US006360215B1

(12) **United States Patent**
**Judd et al.**

(10) **Patent No.: US 6,360,215 B1**
(45) **Date of Patent: Mar. 19, 2002**

(54) **METHOD AND APPARATUS FOR RETRIEVING DOCUMENTS BASED ON INFORMATION OTHER THAN DOCUMENT CONTENT**

(75) Inventors: **Douglass R. Judd**, San Jose; **Paul Gauthier**, San Mateo; **J. Eric Baldeschwieler**, San Francisco, all of CA (US)

(73) Assignee: **Inktomi Corporation**, Foster City, CA (US)

( * ) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/186,058**

(22) Filed: **Nov. 3, 1998**

(51) **Int. Cl.**[7] ..... **G06F 7/00**; G06F 17/30

(52) **U.S. Cl.** ..... **707/3**; 707/4; 707/7; 707/9; 707/10; 707/501; 707/513; 709/202; 382/190

(58) **Field of Search** ..... 707/1, 2, 3, 4, 707/5, 10, 104, 501, 513, 6; 709/201, 203, 218, 219; 382/305

(56) **References Cited**

U.S. PATENT DOCUMENTS

| | | | | |
|---|---|---|---|---|
| 5,465,353 | A * | 11/1995 | Hull et al. | 395/600 |
| 5,757,983 | A * | 5/1998 | Kawaguchi et al. | 382/305 |
| 5,966,710 | A * | 10/1999 | Burrows | 707/103 |
| 6,026,388 | A * | 2/2000 | Liddy et al. | 707/1 |
| 6,065,055 | A * | 5/2000 | Hughes | 709/229 |
| 6,065,056 | A * | 5/2000 | Bradshaw et al. | 709/229 |
| 6,094,649 | A * | 7/2000 | Bowen et al. | 707/3 |
| 6,122,647 | A * | 9/2000 | Horowitz et al. | 707/513 |

* cited by examiner

*Primary Examiner*—Thomas Black
*Assistant Examiner*—Jacques Veillard
(74) *Attorney, Agent, or Firm*—Hickman Palermo Truong & Becker LLP; Edward A. Becker

(57) **ABSTRACT**

A method and apparatus are provided for retrieving documents from a collection of documents based on information other than the contents of a desired document. The collection of documents, which may be a hypertext system or documents available via the World Wide Web, is indexed. In one embodiment, an indexing process of a search engine receives one or more specifications that identify documents, or document locations, and non-content information such as a tag word or code word. The indexing process searches the index to identify all documents in the index that match one or more of the specifications. If a match is found, the tag word is added to the index, and information about the matching document is stored in the index in association with the tag word. A search query is submitted to the search engine. The search query is automatically modified to add a reference to the tag word, such as a query term that will exclude any index entry for a document associated with the tag word. The search is executed against the index, and a set of search results is generated. Accordingly, the search results automatically exclude all documents associated with the tag word. These techniques may be used, for example, to implement a Web search service that produces more accurate search results or that prevents certain documents, such as pornographic materials, from appearing in search results.

**18 Claims, 10 Drawing Sheets**

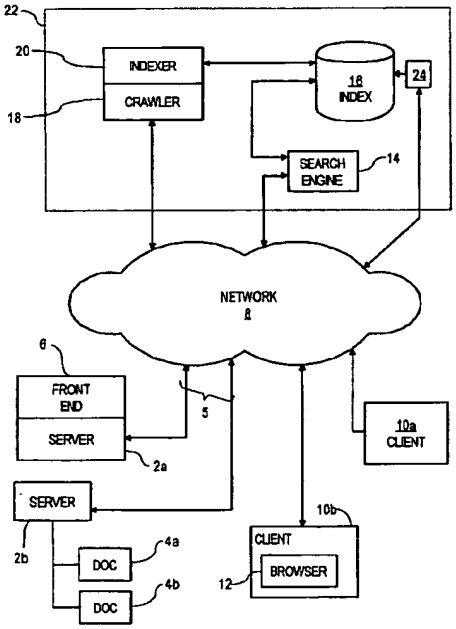


FIG. 1

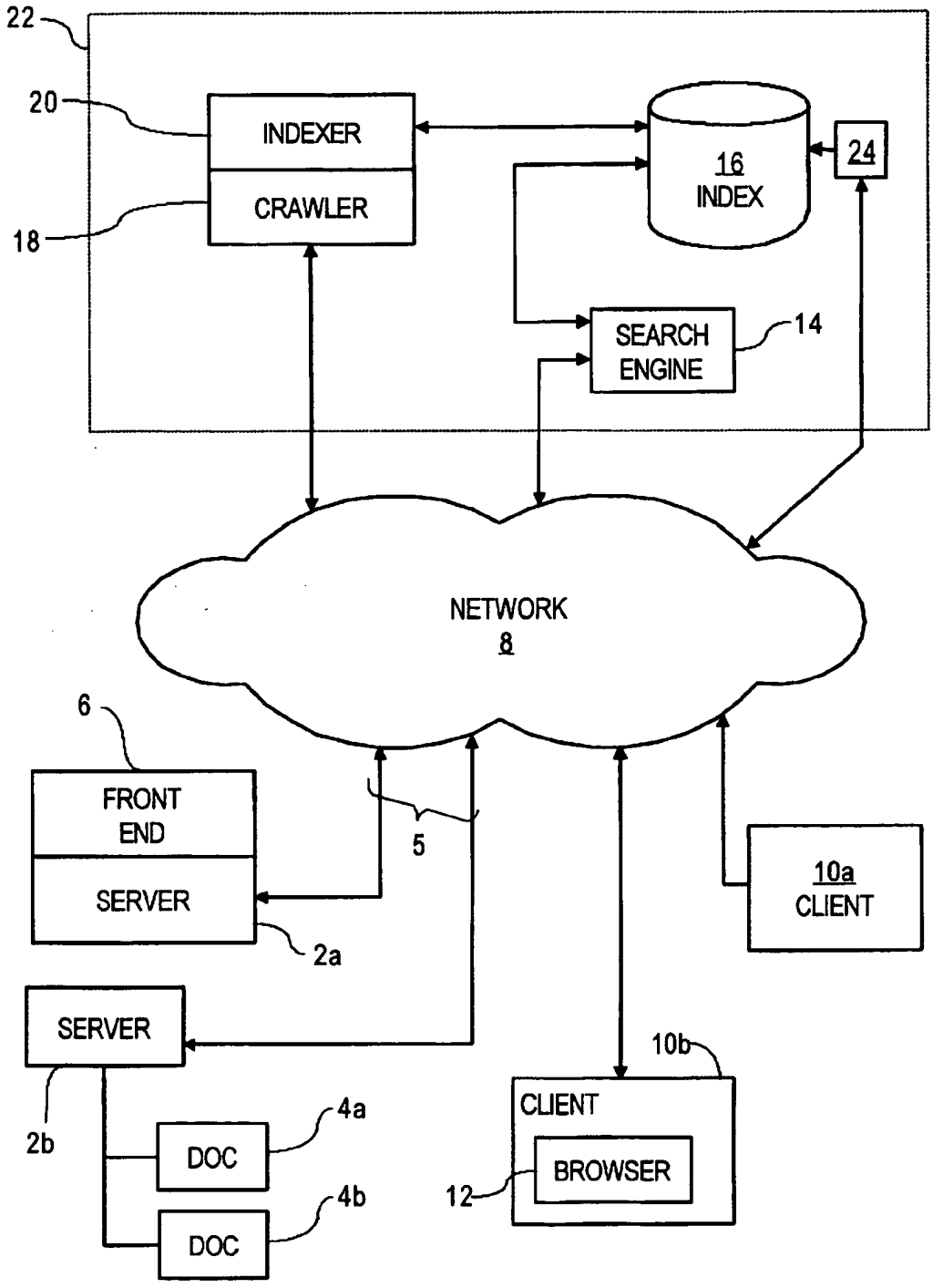

22
20
INDEXER
CRAWLER
18
16
INDEX
24
SEARCH
ENGINE
14
NETWORK
8
6
FRONT
END
SERVER
2a
5
10a
CLIENT
SERVER
2b
10b
CLIENT
BROWSER
12
4a
DOC
4b
DOC

FIG. 2A

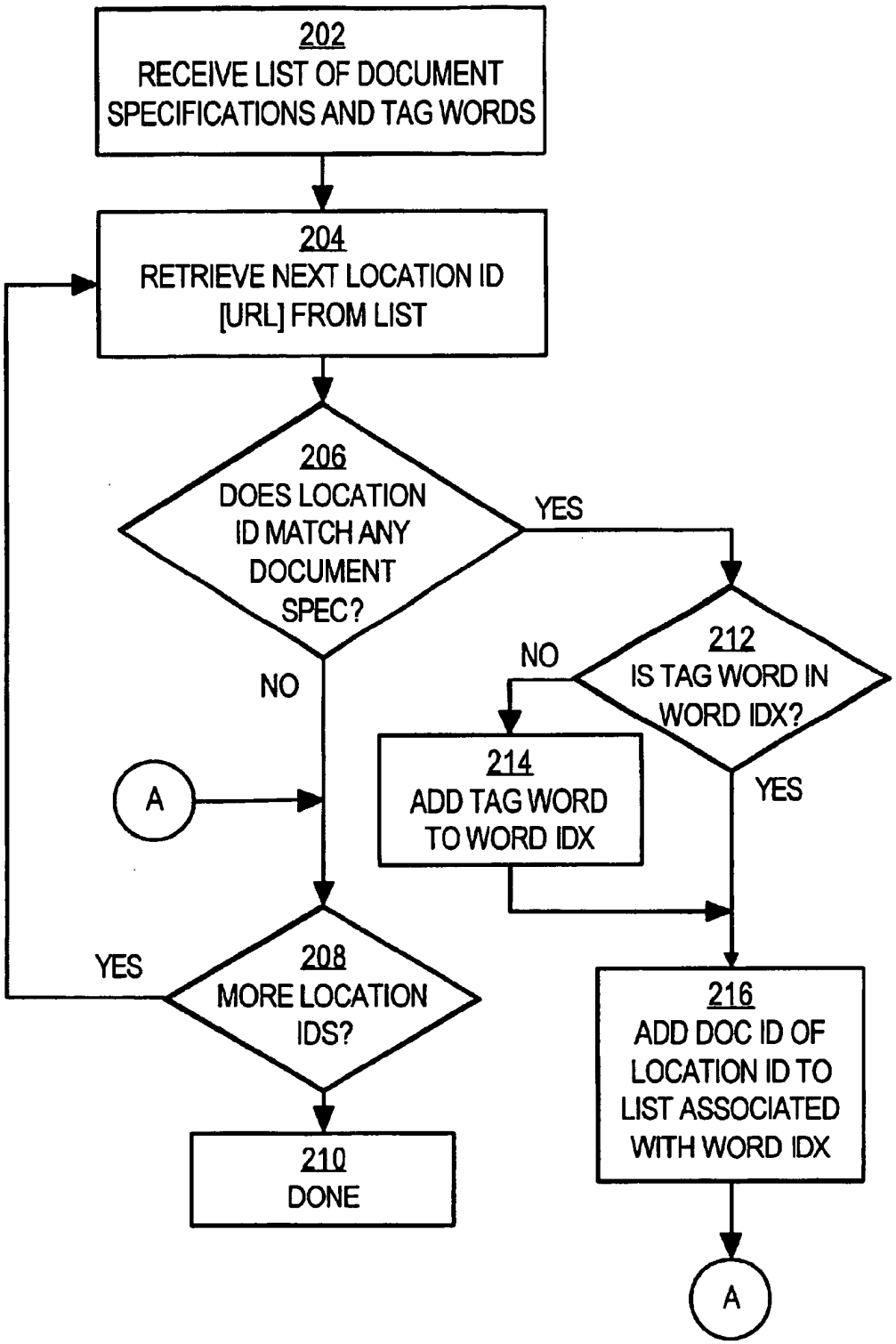

202
RECEIVE LIST OF DOCUMENT SPECIFICATIONS AND TAG WORDS
204
RETRIEVE NEXT LOCATION ID [URL] FROM LIST
206
DOES LOCATION ID MATCH ANY DOCUMENT SPEC?
YES
NO
212
IS TAG WORD IN WORD IDX?
NO
YES
214
ADD TAG WORD TO WORD IDX
A
208
MORE LOCATION IDS?
YES
216
ADD DOC ID OF LOCATION ID TO LIST ASSOCIATED WITH WORD IDX
210
DONE
A

FIG. 2B

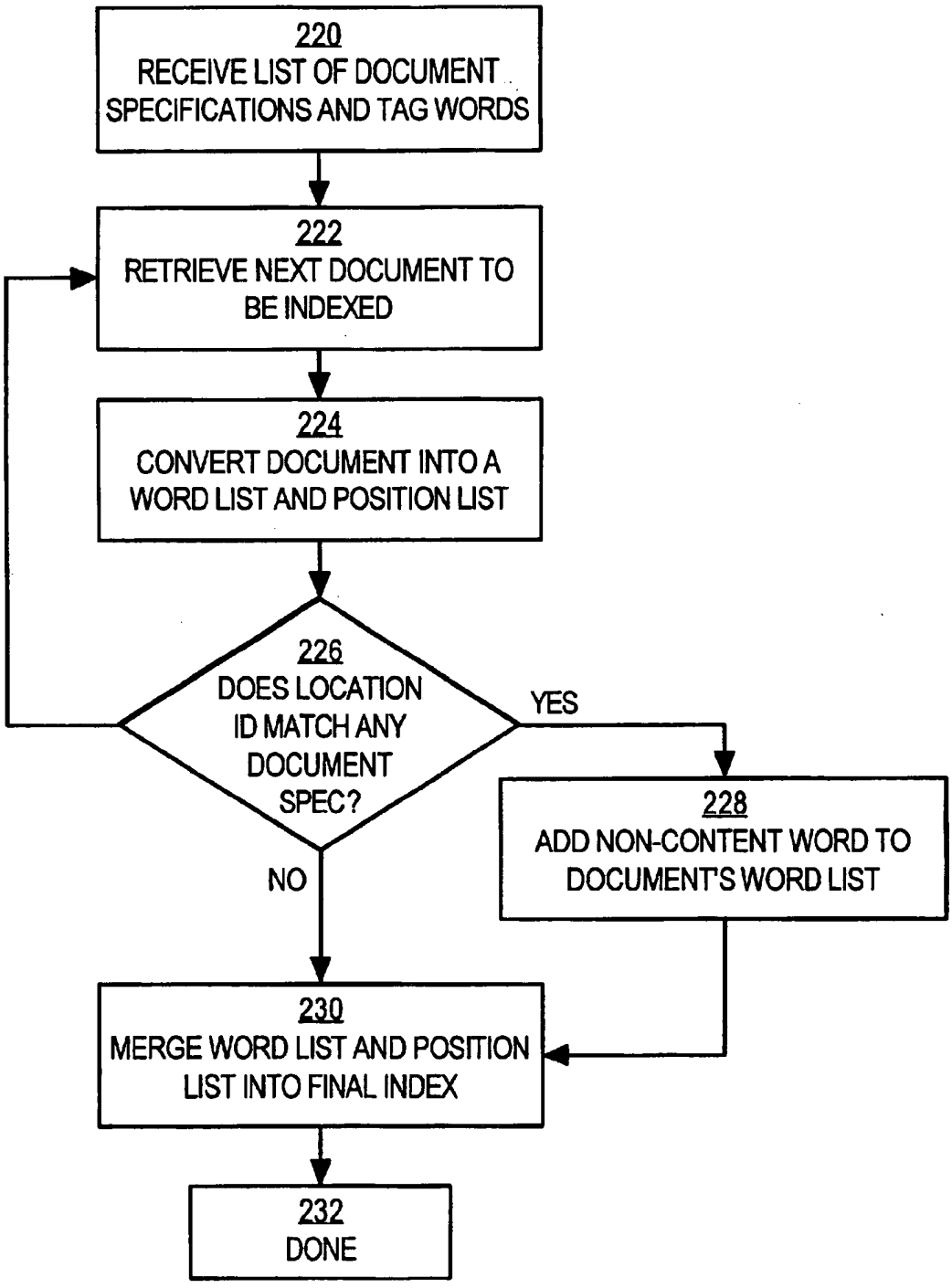
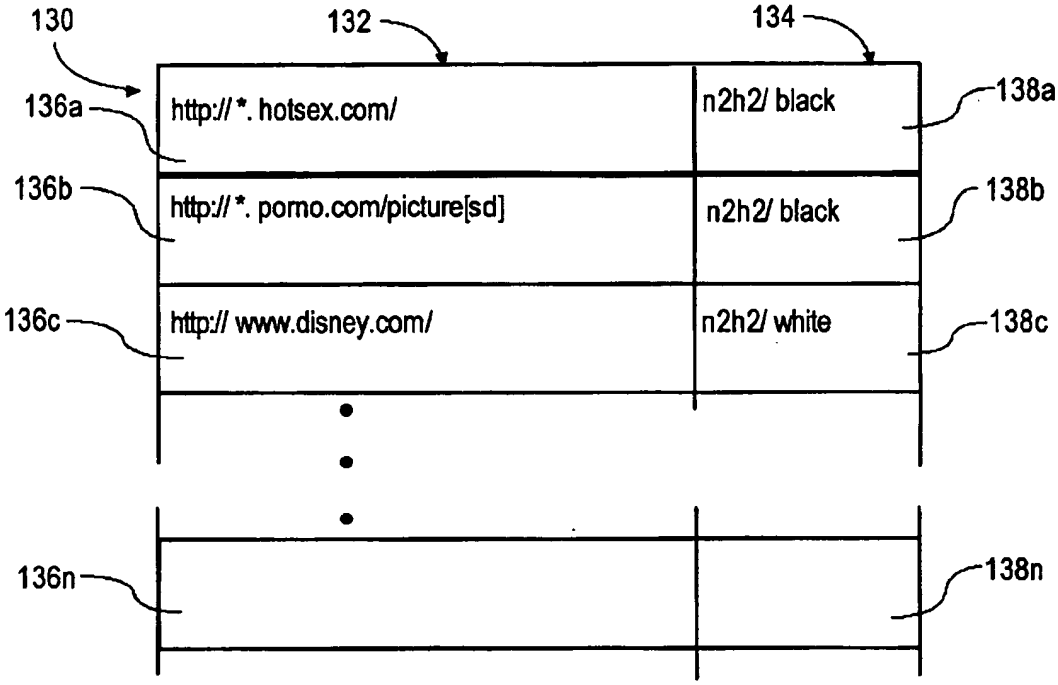

220
RECEIVE LIST OF DOCUMENT SPECIFICATIONS AND TAG WORDS
222
RETRIEVE NEXT DOCUMENT TO BE INDEXED
224
CONVERT DOCUMENT INTO A WORD LIST AND POSITION LIST
226
DOES LOCATION ID MATCH ANY DOCUMENT SPEC?
YES
NO
228
ADD NON-CONTENT WORD TO DOCUMENT'S WORD LIST
230
MERGE WORD LIST AND POSITION LIST INTO FINAL INDEX
232
DONE

FIG. 3


130
132
134
136a
http://*.hotsex.com/
n2h2/ black
138a
136b
http://*.porno.com/picture[sd]
n2h2/ black
138b
136c
http:// www.disney.com/
n2h2/ white
138c
136n
138n

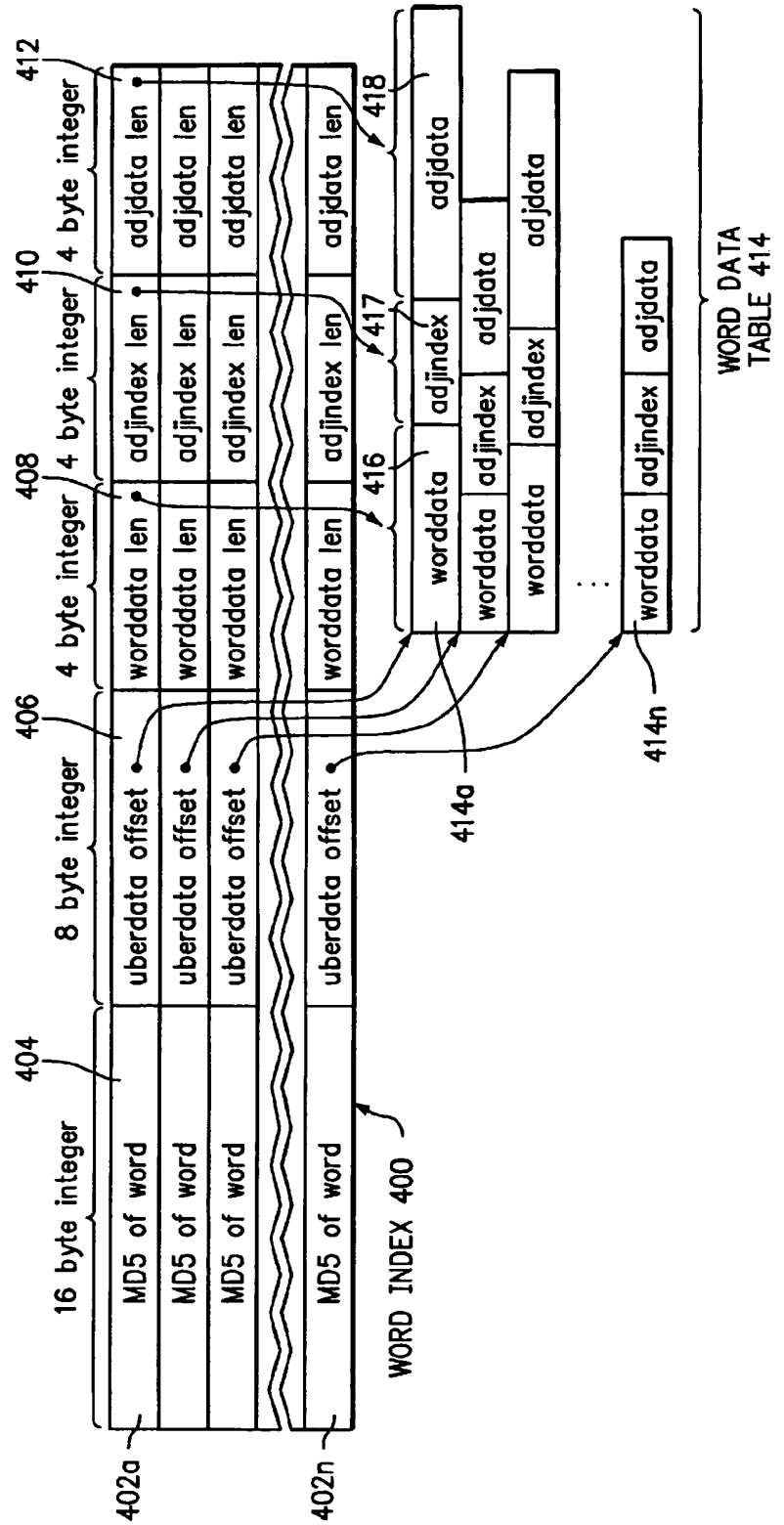

16 byte integer
MD5 of word
8 byte integer
uberdata offset
4 byte integer
worddata len
4 byte integer
adjindex len
4 byte integer
adjdata len
404
406
408
410
412
402a
402n
WORD INDEX 400
416
417
418
414a
414n
worddata
adjindex
adjdata
WORD DATA TABLE 414

FIG. 4A

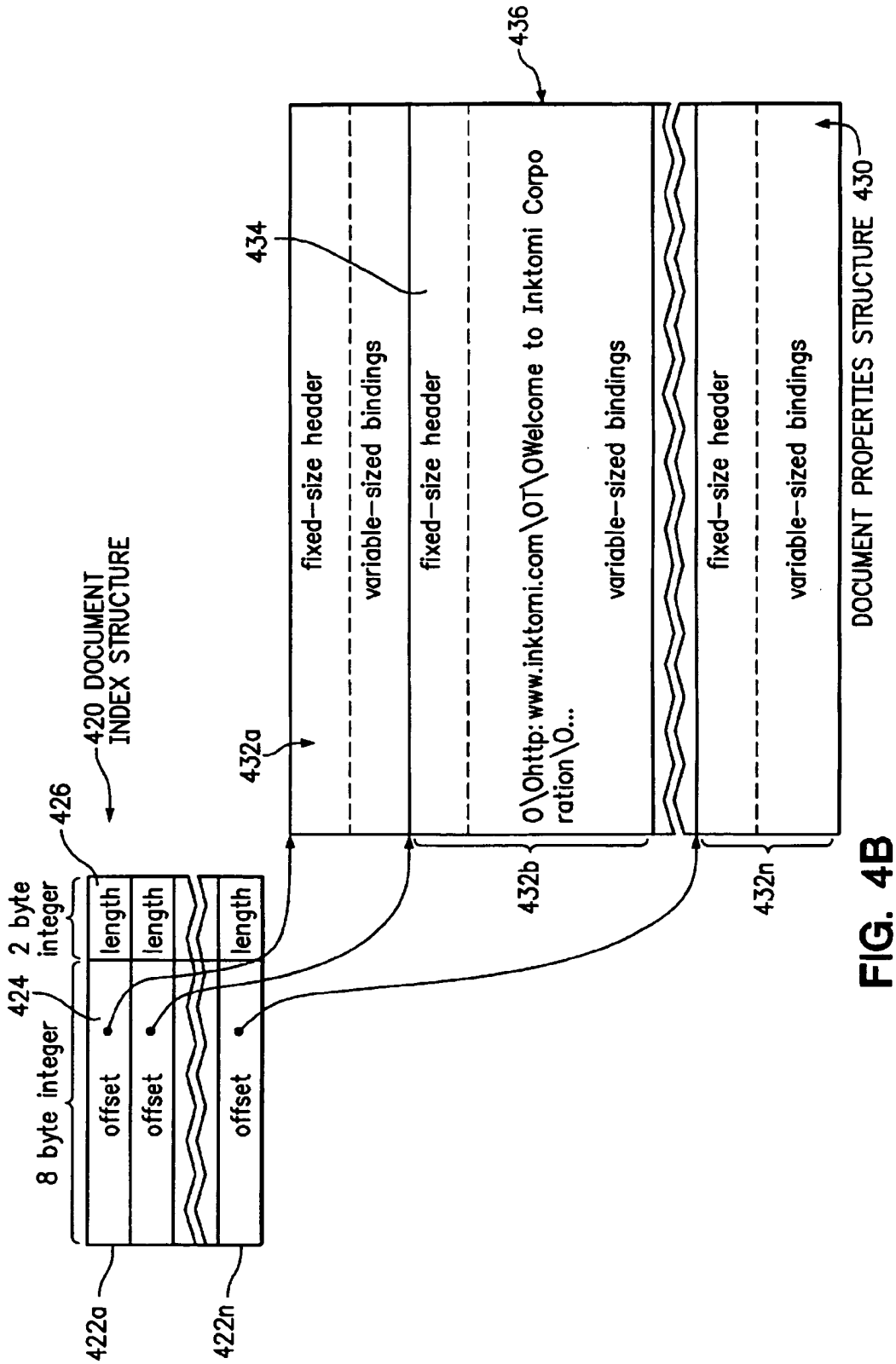

436
434
fixed-size header
variable-sized bindings
fixed-size header
0\Ohttp: www.inktomi.com\OT\OWelcome to Inktomi Corpo
ration\O...
variable-sized bindings
fixed-size header
variable-sized bindings
DOCUMENT PROPERTIES STRUCTURE 430
432a
432b
432n
420 DOCUMENT
INDEX STRUCTURE
426
424 2 byte
integer
8 byte integer
length
length
length
offset
offset
offset
422a
422n

FIG. 4B

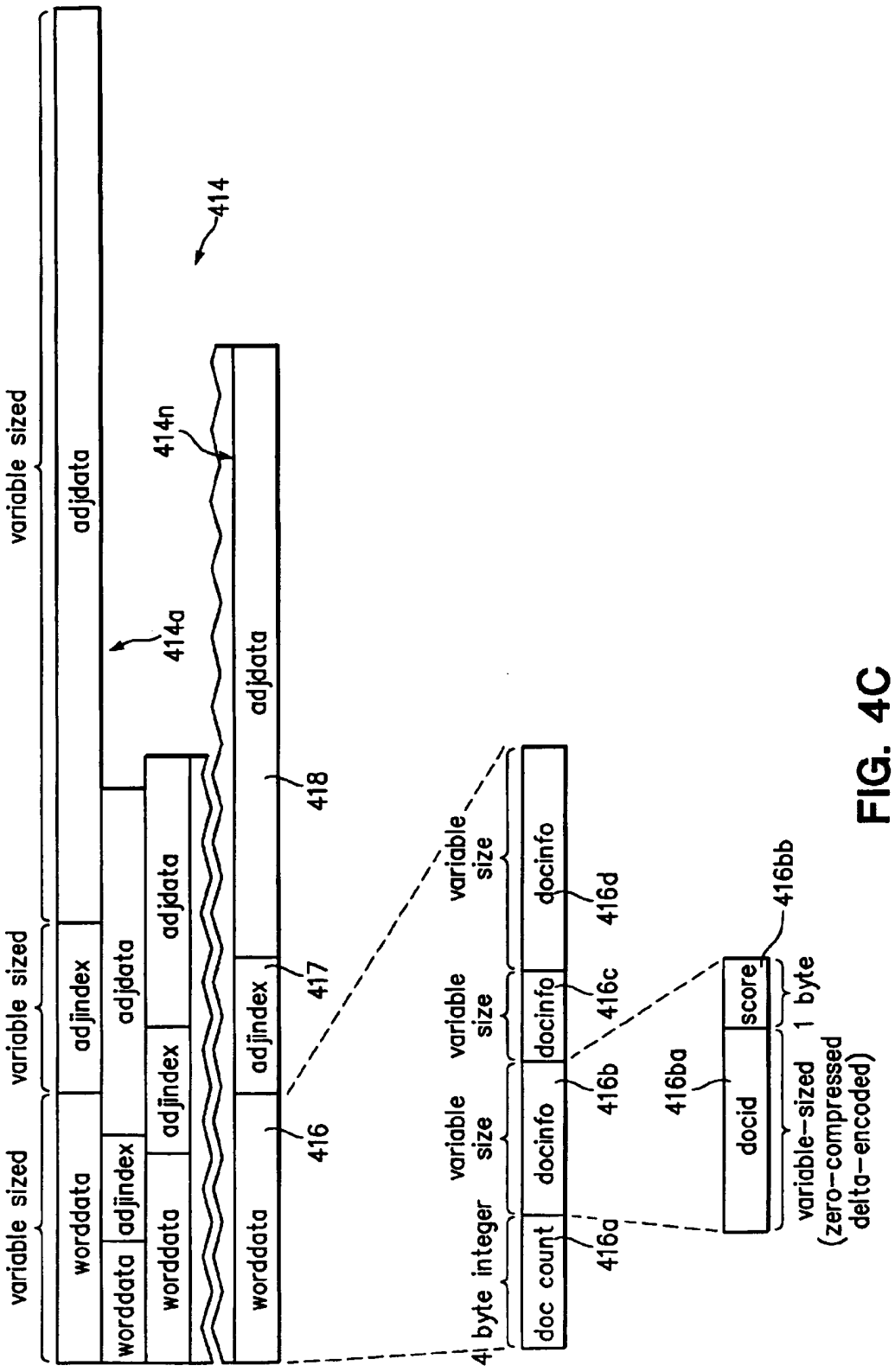

414
414n
414a
variable sized
variable sized
variable sized
worddata
adjindex
adjdata
adjindex
adjdata
adjindex
adjdata
worddata
worddata
adjindex
adjdata
416
417
418
4 byte integer
variable size
doc count
docinfo
variable size
docinfo
variable size
docinfo
416a
416b
416c
416d
416ba
docid
score
416bb
variable-sized
(zero-compressed
delta-encoded)
1 byte

FIG. 4C

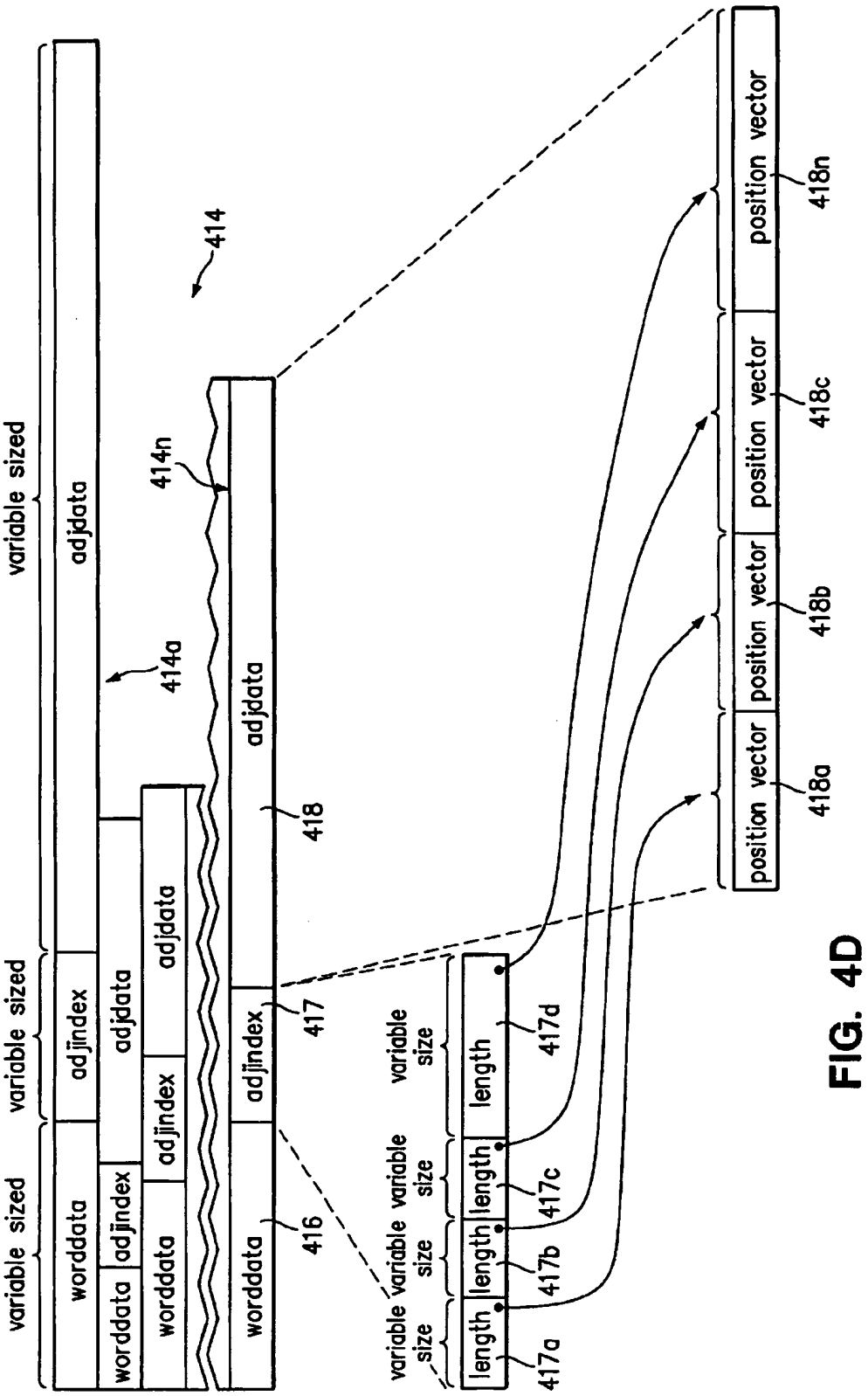
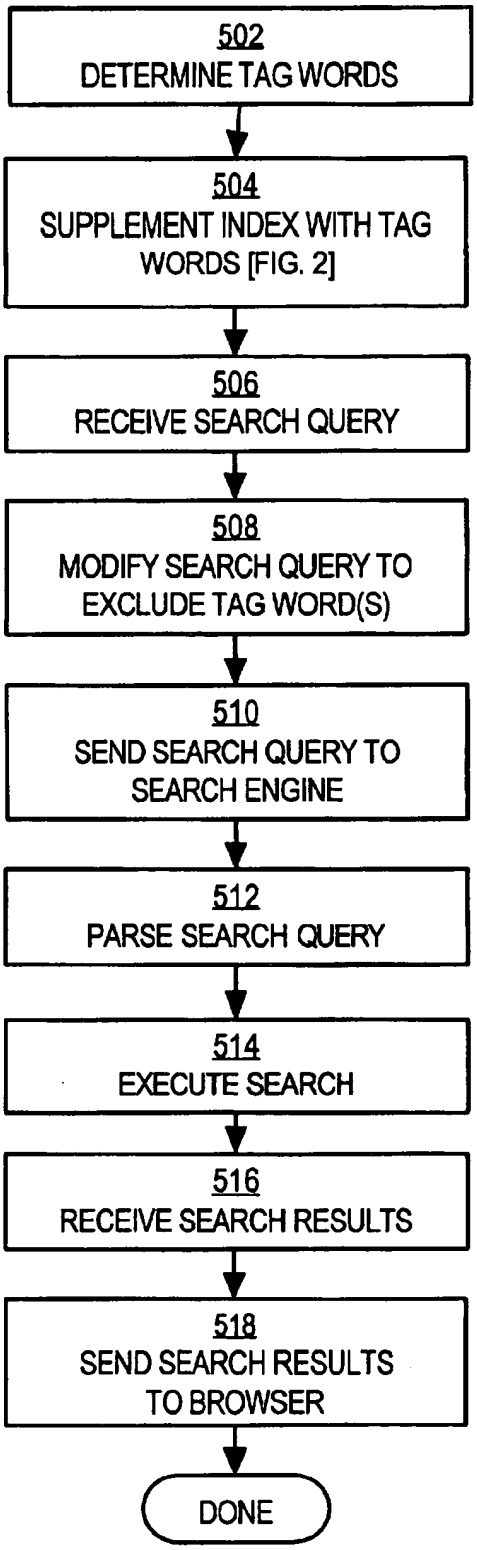

variable sized
variable sized
variable sized
worddata
adjindex
adjdata
414
414n
414a
418
416
417
417a
417b
417c
417d
length
variable size
position vector
418a
418b
418c
418n

FIG. 4D

FIG. 5


502
DETERMINE TAG WORDS
504
SUPPLEMENT INDEX WITH TAG WORDS [FIG. 2]
506
RECEIVE SEARCH QUERY
508
MODIFY SEARCH QUERY TO EXCLUDE TAG WORD(S)
510
SEND SEARCH QUERY TO SEARCH ENGINE
512
PARSE SEARCH QUERY
514
EXECUTE SEARCH
516
RECEIVE SEARCH RESULTS
518
SEND SEARCH RESULTS TO BROWSER
DONE

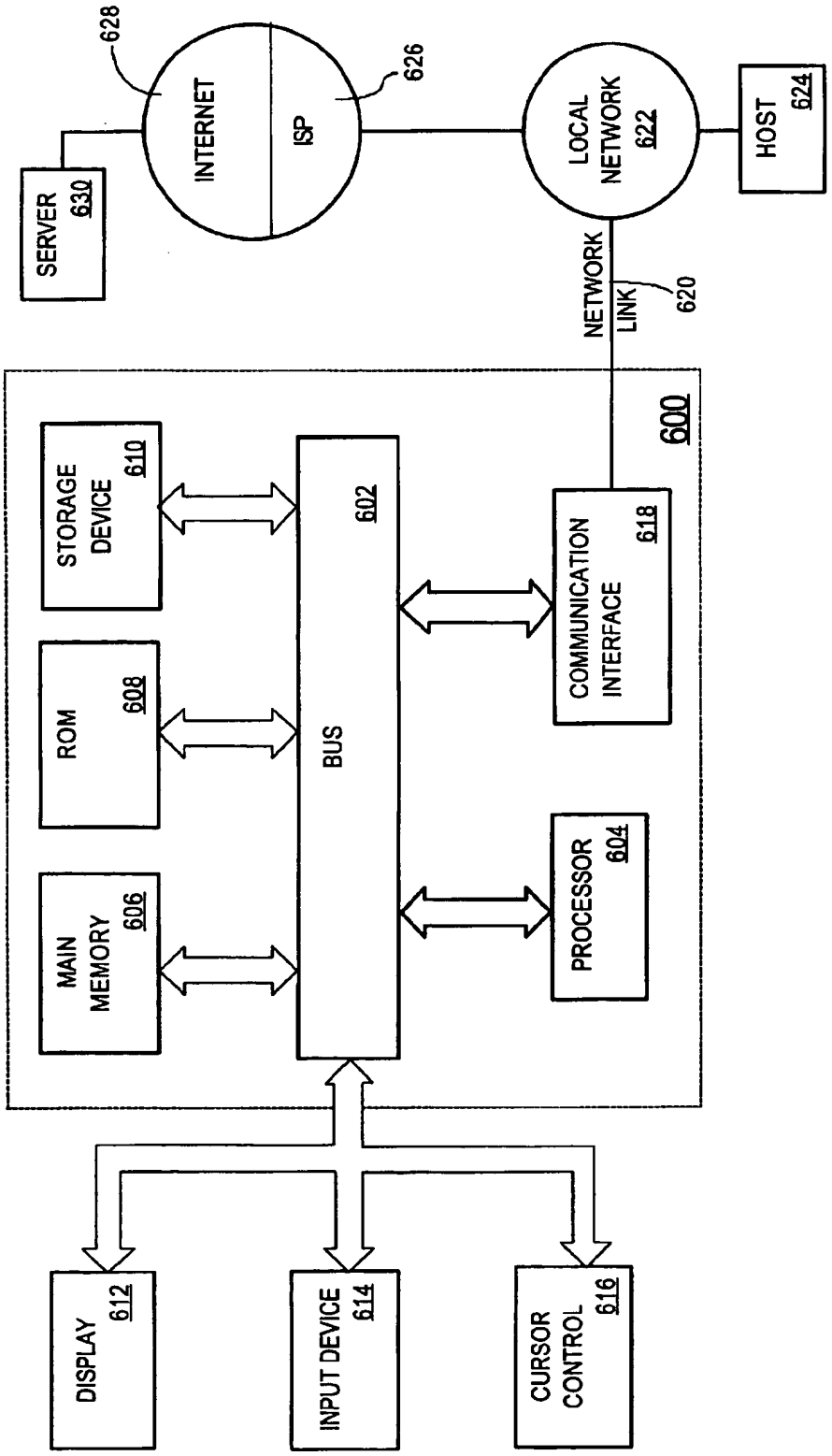

628
INTERNET
ISP
626
LOCAL NETWORK 622
HOST 624
SERVER 630
NETWORK LINK 620
600
STORAGE DEVICE 610
602
COMMUNICATION INTERFACE 618
ROM 608
BUS
MAIN MEMORY 606
PROCESSOR 604
DISPLAY 612
INPUT DEVICE 614
CURSOR CONTROL 616

FIG. 6

# METHOD AND APPARATUS FOR RETRIEVING DOCUMENTS BASED ON INFORMATION OTHER THAN DOCUMENT CONTENT

## FIELD OF THE INVENTION

The present invention generally relates to data processing. The invention relates more specifically to retrieving a document from among several electronic documents based on information not derived from the literal content of the document.

## BACKGROUND OF THE INVENTION

Hypertext systems now enjoy wide use. One particular hypertext system, the World Wide Web ("Web"), provides global access over public packet-switched networks to a large number of hypertext documents. The Web has grown to contain a staggering number of documents, and the number of documents continues to increase. The number of documents available through the Web is so large that to use the Web in a practical way almost always requires a search service, search engine, or similar service.

Certain search engines, however, have limited utility because the search results they produce include documents that are not relevant to the search query. In particular, many search engines return search results that list documents that are not genuinely related to the search query. One reason that search engines return such poor-quality results is that the search engines are easy to deceive. The search engines use "spider" programs that "crawl" to Web servers around the world, locate documents, index the documents, and follow hyperlinks to other documents. The index may comprise a list of all words encountered by the "spider" in all the documents, in which each word in the list is associated with a reference to each of the documents that contains that word. Unfortunately, the "spiders" cannot discriminate among documents that genuinely use a particular word and documents that contain the word, but are really about something else.

For example, a Web document that contains sexually-oriented or pornographic material may also contain one or more words that are unrelated to the sexual material, but are intended to cause the document to be indexed by search engines under those words, thereby luring unsuspecting browsers to the document. A pornographic document that contains a decoy word intended to lure male viewers, such as "CORVETTE," for example, followed by sexual material, would be indexed by a search engine under the word "CORVETTE". The decoy words may be embedded in invisible metatags or rendered in white characters on a white background, so as to be invisible when the document is displayed by the browser. This practice is called "spamming" a search engine or an indexing system. Searchers who submit a query to the search engine or indexing system that seeks information about the motion picture "Bambi" would receive the pornographic page in the search results. This is undesirable and has led to criticism of the utility of search engines and indexing systems.

As a result, the search results returned by the search engine often contain references to the documents that are totally unrelated, in terms of genuine content, to the scope of a search query. In the World Wide Web context, search engines that suffer from this problem include the Yahoo!® Web site, the Excite® Web site, the Infoseek® Web site, and others.

Accordingly, in this field there is a need for a system or mechanism that can eliminate extraneous references from search engine search results.



There is a particular need for a system or mechanism that can combat "spamming" of an indexing system or search engine system.

There is also a need for a mechanism that can associate words, search terms, or editorial matter, other than words appearing in the content of a document, with the document in an index.

There is a particular need for such a system that can carry out a search for a document based on words, search terms, or editorial matter other than the literal content of a group of documents.

## SUMMARY OF THE INVENTION

The foregoing needs, and other needs and objects that will become apparent from the following disclosure, are fulfilled by the present invention, which comprises, in one aspect, a method of selecting electronic documents from among a plurality of electronic documents, the method comprising the steps of storing a tag word in an index in association with information identifying an electronic document, in which the tag word comprises data that is not derived from content of the electronic document; receiving a search query; modifying the search query to create a modified search query by adding to the search query a search term that references the tag word; and creating a set of search results by searching the index based on the modified search query.

One feature of this aspect is that the step of storing includes the steps of receiving data that indicates one or more tag words and criteria to be used to determine which of the plurality of documents should be associated with each of the one or more tag words; and storing, in the index, information associating each of the one or more tag words with the documents in the index that satisfy the criteria associated with the tag words. Another feature is that the step of storing includes the steps of receiving data that indicates one or more tag words and criteria to be used to determine which of the plurality of documents should be associated with each of the one or more tag words, and in which at least a portion of the data is expressed in a wildcard format; retrieving a location identifier of each of the documents that are indexed in the index; matching each location identifier to each of the criteria; and when one location identifier matches one of the criteria, storing, in the index, information associating such location identifier with one or more of the tag words.

In another feature, the step of storing includes the steps of receiving specifications of one or more of the documents that are indexed in the index, in which each of the specifications is associated with one or more tag words, and in which one of the specifications is expressed in a wildcard format; retrieving a location identifier of each of the documents that are indexed in the index; matching each location identifier to each of the specifications by interpreting the one of the specifications that is in the wildcard format according to one or more wildcard format rules; and when one location identifier matches one of the specifications, storing, in the index, information associating such location identifier with one or more of the tag words. In another feature, storing includes the steps of storing a hash value representing the tag word in a record of the index; and storing an indirect reference to information identifying one or more of the documents that contain the tag word.

Another aspect of the invention provides a method of restricting access to an electronic document that is stored among a plurality of documents, the method comprising the steps of storing a tag word in an index in association with

information identifying the electronic document, in which the tag word indicates that access to the electronic document is restricted; receiving a search query that requests the electronic document; modifying the search query to create a modified search query by adding a search term that excludes from the modified search query all documents that contain the tag word; and creating a set of search results by searching the index based on the modified search query. One feature of this aspect is that the step of modifying comprises the step of modifying, automatically and using a software component of a browser, the search query to create a modified search query by adding a search term that excludes from the modified search query all documents that contain the tag word.

Another feature of this aspect is that the modified search query selects only those electronic documents that satisfy the original search query that also contain the tag word. A related feature is that the modified search query selects only those electronic documents that satisfy the original search query that do not contain the tag word.

In another aspect, the invention provides a method of processing queries that select an electronic document from among a plurality of documents, the method comprising the steps of storing a tag word in an index in association with information identifying the electronic document, in which the tag word indicates that access to the electronic document is restricted; receiving a search query that requests the electronic document; modifying the search query to create a modified search query by adding a search term that references the tag word; and creating a set of search results by searching the index based on the modified search query.

One feature of this aspect is that the modifying step further comprises using a software component installed in a browser to perform the steps of intercepting each search query entered using the browser; and modifying the search query that is intercepted to create the modified search query by adding the search term that references the tag word. A related feature is that the step of storing includes the steps of receiving specifications of one or more of the documents that are indexed in the index, in which each of the specifications is associated with the tag word; and storing, in the index, information associating one or more of the documents that are indexed in the index with the tag word, according to the specifications.

Still another aspect of the invention involves a method of constructing an index of a plurality of electronic documents for use in selecting electronic documents from among the plurality of electronic documents, comprising the steps of receiving data that indicates one or more tag words and criteria to be used to determine which of the plurality of documents should be associated with each of the one or more tag words, wherein the tag words are not derived from content of the electronic documents; storing a list of words that are within one document of the plurality of documents; and storing, in the index, information associating each of the one or more tag words with the one document when the one document satisfies the criteria associated with the tag words.

According to yet another aspect, there is a method of constructing an index of a plurality of electronic documents for use in selecting electronic documents from among the plurality of electronic documents, comprising the steps of receiving data that indicates one or more document property values and criteria to be used to determine which of the plurality of documents should be associated with each of the one or more document property values, wherein the document property values are not derived from content of the electronic documents; storing a list of words that are within one document of the plurality of documents; storing, in the index, information associating each of the one or more document property values with the one document when the one document satisfies the criteria associated with the document property values.

Another aspect of the invention is a method of selecting electronic documents from among a plurality of electronic documents, the method comprising the steps of storing a document property value in an index in association with information identifying an electronic document, in which the document property value comprises data that is not derived from content of the electronic document; receiving a search query; modifying the search query to create a modified search query by adding to the search query a search term that references the document property value; and creating a set of search results by searching the index based on the modified search query.

The invention also encompasses a computer system, a computer-readable medium, and a computer data signal embodied in a carrier wave that are configured to carry out the foregoing steps.

The foregoing summary is not intended to describe or summarize all features or aspects of the invention, which are set forth fully in the following description and claims.

## BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

FIG. 1 is a block diagram of a document search system in which one embodiment of the invention may be used.

FIG. 2A is a flowchart of a process of associating non-content information with an index of documents.

FIG. 2B is a flowchart of an alternate process of associating non-content information with an index of documents as the index is constructed

FIG. 3 is a diagram of an exemplary list of document specifications that is received as input by the process of FIG. 2.

FIG. 4A is a diagram of a word index structure that is used in an index organized according to a preferred embodiment of the invention.

FIG. 4B is a diagram of document index structure and a document properties structure that are used in an index organized according to a preferred embodiment of the invention.

FIG. 4C is a diagram of a global data structure and a word data structure that are used in an index organized according to a preferred embodiment of the invention.

FIG. 4D is a diagram of the global data structure of FIG. 4C, and an adjacency index structure, that are used in an index organized according to a preferred embodiment of the invention.

FIG. 5 is a flowchart of a process of carrying out a search query based on non-content information.

FIG. 6 is a block diagram of a computer system hardware arrangement that may be used to carry out an embodiment.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A method and apparatus, for selecting electronic documents from among a plurality of electronic documents, is

described. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

## I. FUNCTIONAL OVERVIEW

Techniques are disclosed for storing, in an index that has been constructed from the content of a plurality documents of a hypertext system, information not found in the documents. These techniques provide a way to associate meta-information, editorial information, commentary, or other tags with documents. Accordingly, the index—and, indirectly, the documents of the hypertext system—can be searched based upon non-content information.

In general, in one embodiment, the foregoing techniques are carried out in the context of an index that has been constructed for a plurality of documents. The index includes an ordered list of words, where each word in the list has been recognized in at least one of the documents. Each word in the list is associated with one or more references to documents that contain that word. The index further includes a list of document location identifiers for all the documents that are referenced in the index. One of the techniques involves receiving a list of document specifications, which may be expressed in literal or wildcard format. A "document specification" is data that indicates criteria for associating documents with a tag word, where the criteria is something other than the documents actually containing the tag word. Each document specification is associated with one or more tag words. Each document location identifier is retrieved from the list and matched against the list of document specifications. If there is a match between the document specification and a document location identifier, then the tag word associated with the document specification is added to the index, and the document associated with the document location identifier that matches the document specification is indexed against that tag word.

As a result, an existing document index is supplemented with tag words, and documents matching one or more of the document specifications are indexed against the tag words in the same way that they would be indexed if they had actually contained the tag words. Accordingly, the index can be searched based upon the tag words, thereby allowing documents in the system to be searched based upon information that is not actually in the documents.

## II. OPERATIONAL CONTEXT

### A. SEARCH SYSTEM

Embodiments of the invention may be implemented in a variety of contexts. A specifically preferred operational context is a search system for the World Wide Web, including a Web crawler or "spider" system, an index of Web documents, and a search engine that can receive a search query and find matching information in the index. The invention and embodiments thereof are not limited to this context, which is illustrated only as an example of how an embodiment can be used.

FIG. **1** is a block diagram of an exemplary operating context for an embodiment that involves an index of a search engine, or similar facility, for set of hypertext documents or for a hypertext database. One or more servers **2***a*, **2***b* are coupled by network connections **5** to a network **8**. The servers **2***a*, **2***b* may be associated with, or store, one or more documents **4***a*, **4***b*, as shown in the case of server **2***b*. Alternatively, a server **2***a* may have a front end component **6**, such as a data presentation layer, that can receive information about documents **4***a*, **4***b* and specially format the information for presentation to a client.

In the preferred embodiment, servers **2***a*, **2***b* are Hypertext Transfer Protocol (HTTP) servers, the network connections **5** are TCP/IP connections or other connections that support HTTP transfers, and the network **8** is the global, packet-switched network known as the Internet.

By way of example, two documents **4***a*, **4***b* are shown. In a practical system, however, there may be millions of documents accessible through thousands of servers. The location of each document is uniquely identified by a location identifier. An example of a location identifier is a Uniform Resource Locator (URL).

One or more clients **10***a*, **10***b* are coupled by network connections to the network **8**. The clients **10***a*, **10***b* are personal computers, workstations, or servers that can request documents **4***a*, **4***b* from the network **8** and present the documents, or information relating to the documents, to a user. In a preferred embodiment, a client **10***b* executes a browser program **12**, such as a World Wide Web browser. The client **10***b* connects to network **8** using a TCP/IP connection, and connects to one of the servers **2***a*, **2***b* using the HTTP protocol. There may be millions of clients **10***a*, **10***b* in a practical embodiment of the system.

In one mode of operation, a client **10***b* submits one or more requests to server **2***b* to retrieve a particular document from among documents **4***a*, **4***b*. The server **2***b* locates the requested document and returns a copy of it to the client **10***b* through the network **8**. A request and response in the HTTP protocol can be used to carry out this mode of operation.

Preferably, a search engine **14** is coupled to the network **8** and to an index **16**. The search engine **14** is a specialized set of one or more software components. The particular internal construction of the search engine **14** is not important, and the structure of a search engine is known in this field. What is important is that the search engine **14** can receive a search request from one of the clients **10***a*, **10***b* or one of the servers ~~**2***a*, **2***b*~~, search the index **16** to identify one or more records that are within the scope of the search request, and return information from the records ("search results") to the requesting client through network **8**.

A crawler **18** is coupled to the network **8** and to an indexer **20**. The crawler **18** and indexer **20** cooperate to periodically visit documents **4***a*, **4***b*, and all other documents that are accessible through the network **8**, and construct the index **16** based on the contents of the documents. The crawler **18** and indexer **20** may comprise software components that carry out the foregoing functions, and may be integrated as one component.

Crawler **18** and indexer **20** may construct and operate on one or more interim indexes that are created offline and made "live" later. An interim index is constructed during an offline crawling and indexing phase, and when the interim index is complete, it is merged into the index **16**. In this way, the crawler **18** and indexer **20** can construct an index without interfering in the operation of search engine **14**, which uses the "live" index **16**. Use of interim indexes is also favored because the indexing process has been found to be memory-intensive. The indexer **20** can build a temporary index in volatile memory, and then store the index information in an interim index in non-volatile memory when the indexer runs out of volatile memory.

### B. CRAWLING AND INDEXING DOCUMENTS

Index **16** may comprise one or more tables, files, or sub-indexes. For example, index **16** may comprise a word

index and a document index. The word index is an alphabetic list of all words encountered by crawler **18** and indexer **20** in all documents **4***a*, **4***b*. Each word in the word index is associated with one or more document identifiers that identify documents that contain the word. The document index maps the document identifiers to specific document location identifiers, or to URLs, or to other information that may be displayed after a search, such as document title or document abstract.

Operation of crawler **18** and indexer **20** may involve receiving and reading one or more document identifiers, each of which identifies a hypertext document. For example, crawler **18** receives URLs, each of which identifies a Web document among documents **4***a*, **4***b*. Crawler **18** may call a process, procedure, program or subroutine and provide it with a list of URLs that the crawler has not yet visited. Crawler **18** may also retrieve a document identified by each URL using an HTTP request.

Each document in the list of URLs is scanned and its content is examined. Each hyperlink within the document is identified. In one embodiment, the documents are formatted using Hypertext Markup Language (HTML), and crawler **18** detects each HTML anchor and associated hypertext reference in the document. The hyperlinks are added to a crawling queue. The crawling queue is a list of document identifiers or URLs that need to be visited by the process. When the process completes processing of the location identifiers that were previously obtained, the process retrieves the next location identifier in the crawling queue. In this way, the process eventually visits all documents to which a particular document points.

For each document that is visited, the system generates a unique document identifier and stores the document identifier in a list of visited documents. In one embodiment, the list is implemented in the form of a vector of visited URLs, in which there is one entry for each visited URL. The process may later search the bit vector to determine whether the system has previously visited a particular URL.

The content of each document is fed to indexer **20**, which carries out two main functions.

First, the indexer **20** constructs an index record of the current document and stores the index record in the document index of index **16**. Each index record contains, among other things, a hash value that uniquely represents the text contents of the associated document. Each index record also contains the location identifier of the current document, and may also contain values of properties that may be displayed in a search result page, such as document title, document summary, or others. The location identifier may also be stored in hashed form.

Second, the indexer **20** reads each word in the document and indexes the document under that word in the word index of index **16**. This function may involve reading a word from a document, checking whether the word is in the word index, adding the word to the word index if it is not found, and associating a reference to the document with the word in the word index. For example, after this process is completed for a set of documents that contain the word "apple", a record of the word index may comprise, in simplified form, the values "apple," "26," "9," "107," "272." This record indicates that the word "apple" appears in documents "26," "9," "107," and "272." The numeric values serve as references to the documents or to the true locations of the documents in the network.

For efficiency and speed, the indexer **16** may store words in the word index in hashed form. For each word in the document, the indexer **16** applies a hash function to the word. In the preferred embodiment, the MD5 hash function is applied, which a fixed-length, 16-byte hash value. The MD5 hash function is described in detail in B. Schneier, "Applied Cryptography" (John Wiley & Sons, 2d ed. 1996), at pp. 436. Use of the MD5 hash function is not required. It is desirable to use a hash function that generates a fixed-length hash value as output, has a uniform distribution of values, and has a low collision rate, such that the hash value uniquely identifies each word that is hashed.

As a result, a rapidly searchable index of all words in all the crawled documents is created. The index is then published to one or more search nodes. Publication may involve sending one or more publication messages to one or more search processes. The publication messages inform the search processes that the sorted index is available for use in searching. The search processes may implement, for example, a Web document search service. An example of a search service that may use the foregoing processes is the HOTBOT® search service commercially available at the URL http://www.hotbot.com/.

C. SEARCHING THE INDEX

To search the index **16**, browser **12** submits a search query to search engine **14**. The search query contains one or more words, for example, "INKTOMI CORPORATION". The search engine **14** matches words in the query to words in the word list of index **16**. The index returns, as search results, information about the documents that are identified by document identifiers associated with matching words in the word list. The returned information may include the title of a document, an abstract, the location identifier or URL of the document. The search results are returned to browser **12** for presentation to a user.

Alternatively, the client **10***a* connects to the server **2***a*. The browser **12** submits a search query to server **2***a*. Server **2***a* forwards the search query to search engine **14** that produces search results as described above. The search results are returned to a server **2***a*. Front end **6** of server **2***a* formats the search results and delivers a formatted page that contains the search results to browser **12**.

## III. ASSOCIATING DOCUMENTS WITH NON-CONTENT INFORMATION

### A. SUPPLEMENTING THE INDEX

FIG. **2A** is a flow diagram of a process of associating non-content information with documents and the index **16**. In this context, "non-content information" refers to any information that does not form part of the literal content of a document, that is, information other than the words or other material of a document that are indexed by the indexer **16** in the manner described above. The non-content information is represented by one or more tag words that are added to the index **16** and associated with one or more documents.

The process of FIG. **2** is undertaken after index **16** is constructed. In particular, the process of FIG. **2** presumes that another process has constructed a word index of the documents that are accessible in a system, and has constructed a list or table of location identifiers or URLs of documents that are accessible to the system.

As shown by block **202**, the process receives a list of records, which may be presented to the process in the form of a data file. Each record comprises a document specification and one or more associated tag words.

FIG. **3** is a block diagram of an exemplary list **130** that comprises document specifications **132** and tag words **134**. List **130** may contain any number of pairs of document specifications and tag words.

A tag word is any character string that is to be associated with a document for search purposes. Often, the tag words are dedicated code words, or words that are not normally found in a document or dictionary, although this characteristic is not required. Examples of tag words include "n2h2/black" and "n2h2/white", as shown by tag words **138***a*, **138***c* of FIG. **3**. Other tag words may be properties or meta-information such as the title of a document, abstract, or others, as described further below. For example, a tag word may be "ADVERTISEMENT" to indicate that its associated Web page(s) contain advertising. A tag word may be "VERIFIED" to indicate that its associated Web page(s) contain factual information that has been verified by some independent third party.

Each of the document specifications **132** specifies a matching criteria. The documents that satisfy the matching criteria are specified in a document specification associated, in the index, with the tag word **134** of the record having that document specification. A document specification **132** may, for example, an expression that identifies the location of a document in a network, such as a URL.

In a preferred embodiment, the document specification may be expressed in a wildcard format. Using a wildcard format for the document specification allows a particular tag word to be associated with more than one URL, without requiring each URL to be identified literally. Document specifications **136***a*–**136***c* of FIG. **3** are expressed in wildcard format. For example, document specification **136***a* is http://*.hotsex.com/. The "*" character in document specification **136***a* indicates that the document specification includes any server within the domain "hotsex.com". When the process of FIG. **2** processes document specification **136***a*, the code word **138***a* will be associated in the index **16** with any indexed document having a location identifier that matches document specification **136***a*, as explained further below.

No particular wildcard format or syntax is required. In practice, however, having a formal wildcard specification or a set of wildcard format rules is advantageous. A preferred wildcard format or syntax is described further below.

The process retrieves the next location identifier or URL in the table of location identifiers of index **16**, as shown by block **204**. The loop formed by block **204** and block **208** represents a sequential retrieval of the location identifier or URL of each document that is indexed in index **16**.

The process then tests whether the current location identifier matches any of the document specifications in the list that was received in block **202**, as shown by block **206**. For example, block **206** may involve matching a URL indexed in index **16** to the document specification **136***a* of list **130**. If there is no match, then control is passed to block **208** to obtain the next location identifier, if any. When document specifications are in wildcard form, block **206** may involve parsing the document specification according to one or more wildcard format rules or syntax rules.

If there is a match, then the process has identified a document indexed in index **16** that is within the scope of a document specification of the list obtained in block **202**. In response, the process retrieves the tag word that is associated with the matching document specification in the list. For example, if the current URL of the index **16** matches document specification **136***a*, then the process retrieves tag word **138***a* from list **130**. The process then determines whether that tag word is currently in the word index of index **16**, as shown by block **212**. If the tag word is not in the word index, then control is passed to block **214**, in which the tag word is inserted into the word index by adding a new record to the word index.

Thereafter, or if the tag word is already in the word index, control is passed to block **216**. The process obtains a document identifier that is associated with the current location identifier. The document identifier is stored in the word index in association with the tag word.

For example, in the context of an index of Web documents, block **206** involves matching a URL indexed in the system to one of the document specifications **132** of list **130**, such as document specification **136***a*. If a match occurs, the process retrieves a document identifier that is associated with the matching URL. The process finds the value of tag word **138***a* in the word index of index **16**. The document identifier is stored in the word index in association with that word value.

As a result, a particular document identified by a particular URL is now indexed in the system in association with a tag word. The process is carried out or iterated for each URL that is indexed in the system and for each of the document specifications **132** in list **130**.

The foregoing describes a process of taking an index after it has been constructed and marking it up with non-content information. Generally, indexing involves taking a document, converting it into a list of words and their positions, and merging these lists into a final index. FIG. **2B** is a flow diagram of an alternate embodiment whereby the non-content information is added to the index at the time the document is being indexed.

As shown in FIG. **2B**, block **220**, a list of document specifications and tag words is received. Block **220** may involve the same steps set forth above in connection with block **202** of FIG. **2A**. In block **222**, the next document to be indexed is retrieved. The steps of block **220** may occur in the normal indexing process in which documents are sequentially retrieved and indexed. Similarly, as part of the normal indexing process, the document is converted into a word list and a list of the positions of the words, as shown in block **224**.

In block **226**, the document's location identifier is compared to the document specifications that were received in block **220**. If a match occurs, then in block **228** the non-content tag words received in block **220** are added to the document's word list. In block **230**, the word list and position lists are merged into the final index. At block **232**, completion of the process yields the complete index.

Thus, in the embodiment of FIG. **2B**, the indexing process is enhanced by doing the tag specification lookup at the time the document is converted into a word list. If any non-content tag words are found, then they are added to the word list before the list is merged into the index.

B. WILDCARD FORMAT FOR DOCUMENT SPECIFICATIONS

Preferably, a document specification in wildcard format may be expressed according to one or more of the following wildcard format rules.

A wildcard may appear in the IP address portion of a document specification. Table 1 compares examples of document specifications having a wildcard designation in the IP portion to the scope or meaning, as interpreted by the process of FIG. **2**, of such specifications.

TABLE 1

WILDCARD IN IP ADDRESS

| DOC SPECIFICATION | SCOPE |
|---|---|
| http://206.19.112.14:8000 | URLs matching this host (port 8000) |
| http://206.19.112.14 | URLs matching this host on default port 80 |

TABLE 1-continued

| WILDCARD IN IP ADDRESS | |
|---|---|
| DOC SPECIFICATION | SCOPE |
| http://206.19.112.[0-9]* | URLs matching this host on any port |
| http://206.19.112.* | URLs on this subnet (port 80) |
| http://206.19.* | URLs on this subnet (port 80) |
| http://206.* | URLs on this subnet (port 80) |

A wildcard may appear in a hostname. Table 2 compares examples of document specifications having a wildcard designation in the hostname to the meaning, as interpreted by the system, of such specifications.

TABLE 2

| WILDCARD IN HOSTNAME | |
|---|---|
| http://www.naughty.psiweb.com:8000 | URLs matching this host (port 8000) |
| http://www.naughty.psiweb.com:[0-9]* | URLs matching this host (any port) |
| http://www.naughty.psiweb.com | URLs matching this host (port 80) |
| http://*.naughty.psiweb.com | URLs matching this subnet (port 80) |
| http://*.psiweb.com | URLs matching this subnet (port 80) |

A wildcard may appear in a path component. If no path component is specified, then any URL that matches the host specification will be tagged. That is, the absence of a path component implies the wildcard designation "/*".

The path component of the URL can be specified using "UNIX-style" filename wildcard patterns. Patterns can be formed with the following elements.

The character "?" matches any single character.

The character "*" matches any sequence of zero or more characters.

The designation "[x . . . y]" matches any single character specified by the set (x . . . y), where any character other than minus sign or close bracket may appear in the set. A minus sign may be used to indicate a range of characters. That is, "[0–5abc]" is a shorthand designation for "[012345abc]". More than one range may appear inside a character set; [0–9a-zA-Z.] matches almost all of the legal characters for a host name.

The designation "[^ x . . . y]" matches any character not in the set x . . . y, which is interpreted as described above for the designation "[x . . . y]".

Some examples of document specifications that have wildcard elements in a path component include:

http://www.childsafe.com/IC*

http://www.netguide.com/part?.html

http://www.nps.gov/fo[a–z]*/*

In the preferred embodiment, except for the special wildcarding characters described above, the rest of the document specification conforms to the URI syntax in the HTTP .1.1 specification, also called Request For Comment (RFC) 2068, which is published at http://www.w3.org/Protocols/rfc2068/rfc2068.

C. TAGGING WITH DOCUMENT PROPERTIES

As noted above, in an alternate embodiment, an indexing system may also be supplemented with non-content document properties. Such supplementation may be done alone or in combination with supplementation by non-content document words.

In this embodiment, the index contains metawords and document properties. "Metawords" are words that documents get indexed against. Documents that contain metawords can be located in the index by combining one or more metawords in a Boolean query and submitting it to the search engine. "Properties", however, cannot be used to locate documents. Properties represent information stored in the index for each document, which can be returned to the client for display on the results page, after the query has been evaluated. Properties include document Title, Abstract, and URL, and or other information that describes a document or its characteristics.

Storage of non-content document properties is useful, for example, to enable a document search system to report descriptive information about documents in association with the results of a search. For example, assume that Company M has a team of people dedicated to evaluating Web pages and writing up small editorial comments on each page. An editorial might say, "This page is full of excellent information and gets a Company M rating of 10." or alternately, "This page gives an adequate description of Sailing in the San Francisco Bay and gets a Company M rating of 6". Company M may want to display this editorial information in its results pages. Company M can achieve this by using the above-described tagging mechanism to associate the non-content editorial information as a document property. The document property information would be stored in the structure illustrated in FIG. 4B.

D. PREFERRED INDEX STRUCTURE

FIG. 4A, FIG. 4B, FIG. 4C, and FIG. 4D are diagrams that show a preferred structure of portions of index **16**, which facilitates association of non-content information with an index of stored documents, and retrieval of documents based on non-content information. Each of the structures described below may be stored in volatile memory and periodically stored in non-volatile storage such as disk storage. Each structure may be organized as a table, file, variable, object, or other data structure defined by an abstract data type in a source program. The index **16** may also include other data structures and program elements that support word indexing, database lookups, and related functions. The particular structure of these elements is not critical. What is important is that the system provides a fast, efficient way to index hypertext documents according to non-content information, and a fast, efficient way to search the index according to a search query and return a set of search results.

FIG. **4A** is a diagram of a preferred embodiment of a word index **400** that comprises one or more records **402***a*–**402***n*. Each record comprises a word hash value **404**, an offset value **406**, a word data length value **408**, an adjacency index length value **410**, and an adjacency data length value **412**. The word hash value **404** is generated by applying a hash function, such as the MD5 function, to a word found in a document. Collectively, the offset value **406**, word data length value **408**, adjacency index length value **410**, and adjacency data length value **412** provide an indirect reference or mapping to records, in a document index, for documents in which the word identified by word hash value **404** appears.

In a preferred embodiment, the offset value **406**, word data length value **408**, adjacency index length value **410**, and adjacency data length value **412** reference a word data table **414**. The word data table **414** comprises a plurality of word data records **414***a*–**414***n*. Each of the word data records **414***a*–**414***n* comprises a word data field **416**, an adjacency index field **417**, and an adjacency data field **418**. Each offset value **406** of a word index record **402***a*–**402***n* points to the

beginning of a record in the word data table **414**. The word data length value **408**, adjacency index length value **410**, and adjacency data length value **412** respectively stored the lengths in bytes of the word data field **416**, adjacency index field **417**, and adjacency data field **418** of the record **414***a*–**414***n* to which the offset value **406** points. These values enable the index **16** to rapidly and efficiently retrieve data from the word data table **414** once a particular word of interest has been identified in the word index **400**. Broadly speaking, the values enable the index **16** to know, in advance of reading the word data table **414**, exactly where to retrieve needed information.

Each word data field **416** stores a "document vector," which is a list of document identifiers in which the associated word appears. Preferably, the document vector is delta-compressed. Collectively, an adjacency index value **417** and an adjacency data value **418** represent a word "position vector" or set of adjacency information. Each adjacency index value **417** stores a set of length values, in which there is one length value for each document in the document vector. Each adjacency data value **418** stores a set of variable-length position vectors. The adjacency information defines what words are located adjacent to a particular word, and is used when searching for word combinations. For example, if a user enters the search query "cat in the hat", the system uses the adjacency information to determine which index entries represent words that are adjacent to another word in the search query. In one embodiment, the adjacency information stores values indicating positions of the word within the document, which values can be used when searching for adjacent word combinations.

FIG. 4C is a diagram of certain internal details of a preferred embodiment of the word data table **414**. Each word data value **416** preferably stores a document count value **416***a*, and one or more document info values **416***b*, **416***c*, **416***d*. The document count value identifies the number of document info values **416***b*, **416***c*, **416***d* that follow. Each document info value summarizes information about the word in a document, and comprises a document identifier **416***ba* and a score value **416***bb*. The document identifier value **416***ba* uniquely identifies a document in the index **16**. For example, a document identifier value **416***ba* of "44" represents the 44$^{th}$ document in the index.

The score value **416***bb* preferably is a one-byte encoded score. In one embodiment, the score value is used to determine how relevant the word is to the document that it is indexed against. For regular document content words, the score value represents the number of times the word appears in the document, normalized over the document's length. For example, if the word appears twenty (20) times in a relatively short document, the score value will be high, whereas if the word appears once in a long document, the score will be low. For tag words, the score may be a fixed value, such as "100". In an alternate embodiment, a user or customer may specify a score for each tag word.

Preferably, the document identifier values **416***ba* are compressed by delta encoding and by using a variable-length coding of the non-zero bytes. Delta encoding involves computing the difference between a current document identifier value and the previous document identifier value. The least significant bit of each byte of a document identifier value **416***ba* is used as a flag that indicates whether another byte follows. The remaining seven (7) bits of each byte together form an integer value, with the least significant bits appearing last. This structure provides a compact and efficient representation of the document identifier values.

FIG. 4D is a diagram of the word data table **414** showing details of the adjacency index values **417**. Each adjacency index value **417** is organized in parallel to the word data value **416** of the same record **414***a*–**414***n*, and has the same number of entries. Each entry is a length value **417***a*–**417***d* and corresponds to a document info value **416***b*, **416***c*, **416***d* of a word data value **416** in the same record **414***a*–**414***n*. Each length value **417***a*–**417***d* represents the length in bytes of a position vector in the adjacency data value **418** of that record. Preferably, each length value **417***a*–**417***d* is delta-encoded and zero-compressed.

This storage scheme is used to improve system performance. Better performance for each query is achieved, in part, by reducing the amount of disk I/O. The vast majority of queries do not require use of adjacency information; only phrase searches, such as "United States", require the adjacency information. By separating the word data from the adjacency data, the system minimizes the amount of data that is read from disk for the typical query. The delta-encoded and zero-compressed values help reduce the amount of space they occupy, further reducing I/O.

Each adjacency data value **418** comprises one or more position vector values **418***a*–**418***n*. Each position vector value **418***a*–**418***n* stores an offset, in bytes, from the beginning of the document, in which the word that is associated with the current record **414***a*–**414***n* appears. Preferably, each position vector value is delta-encoded and zero-compressed.

FIG. 4B is a diagram of a document index structure **420** and a document properties structure **430** that are referenced by the document identifier value **416***ba*. The document index structure **420** provides an indirect mapping of the document identifier value **416***ba* into the document properties structure **430**. The document index structure **420** comprises a plurality of records **422***a*–**422***n*, in which each record has an offset value **424** and a length value **426**. The document identifier value **416***ba* of the word data value **416** is equivalent to the relative position of records in the document index structure **420**. Thus, a document identifier value **416***ba* having a value of "4" references the fourth record **422***a*–**422***n* of the document index structure **420**.

The document properties structure **430** comprises a plurality of record pairs **432***a*–**432***n*. Each record pair comprises a fixed-size header **434** and a variable-size bindings section **436**. Headers **434** store static document information such as the time the document was crawled or scanned, format, document length, etc. The bindings sections **436** each store document property information in the form of one or more tag/value pairs. Tags and values are null-terminated strings. Tags are one- or two-character mnemonics. Values are text strings that represent a property value. For example, the tag "U" means "URL", and an associated value might be "http:www.inktomi.com".

Each offset value **424** of the document index structure **420** specifies or points to a relative offset of the document properties structure **430**. Each length value **426** specifies the length in bytes of the record pair **432***a*–**432***n* to which an associated offset value **424** points.

Using this structure, a particular document may be rapidly and efficiently located based upon a word that appears in the document. Words are indexed in the word index **400**. A particular word of interest is found in the word index **400** using a search based on hash value. Information in a matching record of the word index record **400** points to a record of the word data table **414**. A document identifier value **416***ba* in the record of the word data table points to the document index structure **420**. The document index structure **420** points to a record of the document properties structure **430**. Values in the document properties structure **430** specifically identify a document that contains the

matched word. The values can be provided to a browser, or the referenced document can be retrieved.

Other support structures and files for index **16** may be provided. For example, index **16** may include a word list ("lexdata") file that contains a list of all words in the database. Index **16** may include a database types ("dbtypes") file that contains a list of database identifiers for the index. It may be used by the search engine **14** to allow searches that are restricted to a subset of cluster nodes.

A "cluster" is a group of tightly coupled workstations used to achieve parallelism, fault-tolerance and scalability. Each workstation in the cluster is called a "node". In a preferred embodiment, the word index is distributed across the nodes in a cluster of workstations. Each workstation holds the index information for a subset of the World Wide Web. The aggregate index is called a "database".

Index **16** also may include a deleted document list ("docid" file) that stores a list of deleted documents, represented by a list of document identifiers and, optionally, a location identifier or URL. Index **16** may also have a server info list ("serverInfo" file) that stores a record for each unique server that the indexed documents came from. Each record may store a server type value and server IP address value derived from the "Server" header of an HTTP response that is received in response to a request to retrieve a document from that server. Index **16** may also contain a version file that stores information identifying the current version of the index and its constituent files, to enable proper synchronization.

## IV. SEARCHING BASED ON NON-CONTENT INFORMATION

In one mode of use, the techniques disclosed herein can be used in a variety of useful document search applications.

FIG, **5** is a flow diagram showing a process of searching an index, which has been tagged in the manner described above, to provide a service that filters Web documents to eliminate documents available over the Web that are considered undesirable for viewing or review by children.

In block **502**, tag words to identify desirable or undesirable documents are determined. Block **502** may involve coining or deciding upon appropriate tag words for the service to be provided. There may be one, two, or more tag words. For example, the tag words are "ACME/BAD" and "ACME/GOOD". The tag words indicate, respectively, that Acme Corporation has reviewed a particular page and determined whether it is good or bad for children.

The index **16** is supplemented by associating documents in the index with one or more of the tag words, as indicated in block **504**. Assume, for purposes of this example, that all bad documents containing the word SEX have been indexed in the index **16** in association with the tag word "ACME/BAD." This step may be carried out using the process of FIG. **2**. For example, Acme provides to index **16** a list of document specifications that identify bad Web documents containing the word SEX and that are to be associated with tag word "ACME/BAD." There may be documents indexed in the index that contain the word SEX but are not deemed bad, and which are not indexed in association with the tag word.

A search query is formulated and received by the process, as shown by block **506**. For example, suppose a child at the workstation enters the search query "SEX". Internally, this query is represented in the format "+SEX". The "+" character reflects Boolean AND logic. Thus, the character string "+SEX" means "find documents that contain the word SEX."

In block **508**, the search query is modified in a manner that excludes one or more of the tag words from the scope of the search query. For example, the browser **12** intercepts the search query and adds the command "−ACME/BAD" to the query. The "−" character is reflects Boolean NOT logic. Thus, the query will omit any document that is indexed under "ACME/BAD". In one embodiment, Acme provides a browser plug-in program that the user installs in association with the browser **12**, and the plug-in program intercepts the search query and modifies it. Alternatively, browser **12** has a built-in process that modifies search queries. According to another alternative, multiple different search engines use the same search directory, but apply different filters to the search query. This allows the search results to be tailored to a particular audience that is served by each search engine. For example, a search engine intended for an audience of professionals in the medical field would apply a filter that includes documents of interest to the medical community in the search results. According to yet another alternative, a user may select the filters that the user wishes to use in the query, and store the filters in a "cookie" file in association with the browser. In this alternative, the search engine is tailored personally to the preferences of an individual user.

Other alternatives—(1) different search engines use the same search directory but apply different filters. This allows audience-tailored search engines log a kid-safe search origin, (2) let the user select the files he wants to and store them in a cookie to personally tailor the search engine.

The browser sends an HTTP request, containing the modified search query, to the search engine **14**, as shown by block **510**. At block **512**, the search engine **14** parses the search request to determine its meaning and how to query the index **16** properly. At block **514**, search engine **14** executes a search against the index **16**, which has been supplemented with the tag words as described above. In block **516**, search engine **14** receives a set of search results from the index **16**. Because of the manner in which the search query has been modified, all documents indexed under the tag word "ACME/BAD" are excluded from the search results. In block **518**, the search engine **14** delivers the search results to the browser **12**.

As a result, browser **12** receives a filtered set of search results in which documents deemed bad for children have been removed. The child does not see Web documents that have been indexed under "ACME/BAD."

In another embodiment, the documents contain actual or allegedly factual information, and the tag words indicate whether an impartial third party has verified the truth the information. The tag words are used to limit a Web search to only those documents that have been verified by the third party.

In still another embodiment, a tag word indicates whether a document contains advertising. The tag word is used to formulate a search query that will filter out advertising from the search results.

Thus, embodiments of the invention are applicable to any context in which a third party labels Web documents with a label that contains meta-information or other descriptive information.

Another advantage is that, since the tags are indexed just as if they were words contained in the document, no special modifications have to be made to the indexing or search logic in order to support non-content based filtering.

Further, embodiments of the invention may be used to implement a service that can filter out, from a set of search results, documents that contain a particular search term but

are not about that search term. For example, consider a search query submitted by a dog aficionado that comprises the word "DOGS". The tag word is "AKC/DOGS." The American Kennel Club, which is a respected authority on dogs in the United States, provides a list of specifications which, in their simplest form, may be URLS, that correspond to documents that the AKC has verified to be genuinely about dogs. The search query is automatically converted to a query in the form "AKC/DOGS". Thus, the search query retrieves only documents about dogs that have been reviewed and approved by the American Kennel Club. Moreover, documents that contain the word "DOGS" but that have nothing to do with dogs are filtered out of the search results. Thus, the invention provides an effective weapon against "spamming" of an indexing system.

In another embodiment, a search service comprises one or more search nodes and one or more master nodes. Multiple search nodes are used to distribute search loading and to distribute index loading. Collectively, the search nodes and master node are called "back end" elements. In the preferred embodiment, one or more separate servers are "front end" elements. The front-end elements provide an interface to a browser. For example, the front-end elements may format the search results and may present the search results to the browser in a custom or specialized manner. An example of such presentation is a customized result page prepared using an HTML template.

One of the master nodes accepts an HTTP connection and receives a search request from a browser or from another processor. The master node broadcasts the search request to all nodes of the search system. Each node contains a search engine that can open a socket connection to the index, including the structures described above. Each node executes the search request against the index structures. Each node returns a set of search results to the master node. The master node merges all search results and returns the merged search results to the client. When the master node merges the search results, the master node reads search query and applies it to the search results, thereby adding or eliminating documents that are indexed under a particular tag word that appears in the search request. The merged search results, filtered according to the request and the tag word, are sent to the front-end elements. The front-end elements format the search results and display them at the browser.

## V. HARDWARE OVERVIEW

FIG. **6** is a block diagram that illustrates a computer system **600** upon which an embodiment of the invention may be implemented. Computer system **600** includes a bus **602** or other communication mechanism for communicating information, and a processor **604** coupled with bus **602** for processing information. Computer system **600** also includes a main memory **606**, such as a random access memory (RAM) or other dynamic storage device, coupled to bus **602** for storing information and instructions to be executed by processor **604**. Main memory **606** also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor **604**. Computer system **600** further includes a read only memory (ROM) **608** or other static storage device coupled to bus **602** for storing static information and instructions for processor **604**. A storage device **610**, such as a magnetic disk or optical disk, is provided and coupled to bus **602** for storing information and instructions.

Computer system **600** may be coupled via bus **602** to a display **612**, such as a cathode ray tube (CRT), for displaying information to a computer user. An input device **614**, including alphanumeric and other keys, is coupled to bus **602** for communicating information and command selections to processor **604**. Another type of user input device is cursor control **616**, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor **604** and for controlling cursor movement on display **612**. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane.

The invention is related to the use of computer system **600** for selecting electronic documents from among a plurality of such documents. According to one embodiment of the invention, selecting electronic is provided by computer system **600** in response to processor **604** executing one or more sequences of one or more instructions contained in main memory **606**. Such instructions may be read into main memory **606** from another computer-readable medium, such as storage device **610**. Execution of the sequences of instructions contained in main memory **606** causes processor **604** to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions to processor **604** for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device **610**. Volatile media includes dynamic memory, such as main memory **606**. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus **602**. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punchcards, papertape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

Various forms of computer readable media may be involved in carrying one or more sequences of one or more instructions to processor **604** for execution. For example, the instructions may initially be carried on a magnetic disk of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system **600** can receive the data on the telephone line and use an infrared transmitter to convert the data to an infrared signal. An infrared detector can receive the data carried in the infrared signal and appropriate circuitry can place the data on bus **602**. Bus **602** carries the data to main memory **606**, from which processor **604** retrieves and executes the instructions. The instructions received by main memory **606** may optionally be stored on storage device **610** either before or after execution by processor **604**.

Computer system **600** also includes a communication interface **618** coupled to bus **602**. Communication interface

618 provides a two-way data communication coupling to a network link 620 that is connected to a local network 622. For example, communication interface 618 may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface 618 may be a local area network (LAN) card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface 618 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

Network link 620 typically provides data communication through one or more networks to other data devices. For example, network link 620 may provide a connection through local network 622 to a host computer 624 or to data equipment operated by an Internet Service Provider (ISP) 626. ISP 626 in turn provides data communication services through the worldwide packet data communication network now commonly referred to as the "Internet" 628. Local network 622 and Internet 628 both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link 620 and through communication interface 618, which carry the digital data to and from computer system 600, are exemplary forms of carrier waves transporting the information.

Computer system 600 can send messages and receive data, including program code, through the network(s), network link 620 and communication interface 618. In the Internet example, a server 630 might transmit a requested code for an application program through Internet 628, ISP 626, local network 622 and communication interface 618. In accordance with the invention, one such downloaded application provides for selecting electronic documents as described herein.

Processor 604 may execute the received code as it is received, and/or stored in storage device 610, or other non-volatile storage for later execution. In this manner, computer system 600 may obtain application code in the form of a carrier wave.

## EXTENSIONS AND ALTERNATIVES

In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

What is claimed is:

1. A method of selecting electronic documents from among a plurality of electronic documents, the method comprising the steps of:

storing a tag word in an index in association with information identifying an electronic document, in which the tag word comprises data that does not appear in a content of the electronic document;

receiving a search query;

modifying the search query to create a modified search query by adding to the search query a search term that references the tag word; and

creating a set of search results by searching the index based on the modified search query;

wherein the step of storing includes the steps of:

receiving data that indicates one or more tag words and criteria to be used to determine which of the plurality of documents should be associated with each of the one or more tag words, and in which at least a portion of the data is expressed in a wildcard format;

retrieving a location identifier of each of the documents that are indexed in the index;

matching each location identifier to each of the criteria; and

when one location identifier matches one of the criteria, storing, in the index, information associating such location identifier with one or more of the tag words.

2. A method of selecting electronic documents from among a plurality of electronic documents, the method comprising the steps of:

storing a tag word in an index in association with information identifying an electronic document, in which the tag word comprises data that does not appear in a content of the electronic document;

receiving a search query;

modifying the search query to create a modified search query by adding to the search query a search term that references the tag word; and

creating a set of search results by searching the index based on the modified search query;

wherein the step of storing includes the steps of:

receiving specifications of one or more of the documents that are indexed in the index, in which each of the specifications is associated with one or more tag words, and in which one of the specifications is expressed in a wildcard format;

retrieving a location identifier of each of the documents that are indexed in the index;

matching each location identifier to each of the specifications by interpreting the one of the specifications that is in the wildcard format according to one or more wildcard format rules; and

when one location identifier matches one of the specifications, storing, in the index, information associating such location identifier with one or more of the tag words.

3. A method of processing queries that select an electronic document from among a plurality of documents, the method comprising the steps of:

storing a tag word in an index in association with information identifying the electronic document, in which the tag word indicates that access to the electronic document is restricted;

receiving a search query that requests the electronic document;

modifying the search query to create a modified search query by adding a search term that references the tag word; and

creating a set of search results by searching the index based on the modified search query;

wherein the step of storing further includes the steps of:

receiving specifications of one or more of the documents that are indexed in the index, in which each of the specifications is associated with the tag word, and in which each of the specifications is expressed in a wildcard format;

retrieving a location identifier of each of the documents that are indexed in the index;

matching each location identifier to each of the specifications; and

when one location identifier matches one of the specifications, storing, in the index, information associating such location identifier with the tag word.

4. A method of processing queries that select an electronic document from among a plurality of documents, the method comprising the steps of:

storing a tag word in an index in association with information identifying the electronic document, in which the tag word indicates that access to the electronic document is restricted;

receiving a search query that requests the electronic document;

modifying the search query to create a modified search query by adding a search term that references the tag word; and

creating a set of search results by searching the index based on the modified search query;

wherein the step of storing includes the steps of:

receiving specifications of one or more of the documents that are indexed in the index, in which each of the specifications is associated with the tag word, and in which one of the specifications is expressed in a wildcard format;

retrieving a location identifier of each of the documents that are indexed in the index;

matching each location identifier to each of the specifications by interpreting the one of the specifications that is in the wildcard format according to one or more wildcard format rules; and

when one location identifier matches one of the specifications, storing, in the index, information associating such location identifier with the tag words.

5. A method of processing queries that select an electronic document from among a plurality of documents, the method comprising the steps of:

storing a tag word in an index in association with information identifying the electronic document, in which the tag word indicates that access to the electronic document is restricted;

receiving a search query that requests the electronic document;

modifying the search query to create a modified search query by adding a search term that references the tag word; and

creating a set of search results by searching the index based on the modified search query;

wherein the step of storing includes the steps of:

receiving data that indicates one or more tag words and criteria to be used to determine which of the plurality of documents should be associated with each of the one or more tag words, and in which at least a portion of the data is expressed in a wildeard format;

retrieving a location identifier of each of the documents that are indexed in the index;

matching each location identifier to each of the criteria; and

when one location identifier matches one of the criteria, storing, in the index, information associating such location identifier with one or more of the tag words.

6. A method of constructing an index of a plurality of electronic documents for use in selecting electronic documents from among the plurality of electronic documents, the method comprising the steps of:

receiving data that indicates one or more tag words and criteria to be used to determine which of the plurality of documents should be associated with each of the one or more tag words, wherein the tag words do not appear in a content of the electronic documents;

storing a list of words that are within one document of the plurality of documents;

storing, in the index, information associating each of the one or more tag words with the one document when the one document satisfies the criteria associated with the tag words;

wherein the step of receiving data includes the steps of receiving data that indicates one or more tag words and criteria to be used to determine which of the plurality of documents should be associated with each of the one or more tag words, and in which at least a portion of the data is expressed in a wildcard format; and

wherein the step of storing information comprises the steps of retrieving a location identifier of each of the documents; matching each location identifier to each of the criteria; and when one location identifier matches one of the criteria, storing, in the index, information associating such location identifier with one or more of the tag words.

7. A method of constructing an index of a plurality of electronic documents for use in selecting electronic documents from among the plurality of electronic documents, the method comprising the steps of:

receiving data that indicates one or more tag words and criteria to be used to determine which of the plurality of documents should be associated with each of the one or more tag words, wherein the tag words do not appear in a content of the electronic documents;

storing a list of words that are within one document of the plurality of documents;

storing, in the index, information associating each of the one or more tag words with the one document when the one document satisfies the criteria associated with the tag words;

wherein the step of receiving data includes the steps of receiving specifications of one or more of the documents that are indexed in the index, in which each of the specifications is associated with one or more tag words, and in which one of the specifications is expressed in a wildcard format;

and wherein the step of storing information comprises the steps of:

retrieving a location identifier of each of the documents that are indexed in the index;

matching each location identifier to each of the specifications by interpreting the one of the specifications that is in the wildeard format according to one or more wildcard format rules; and

when one location identifier matches one of the specifications, storing, in the index, information associating such location identifier with one or more of the tag words.

8. A method of constructing an index of a plurality of electronic documents for use in selecting electronic documents from among the plurality of electronic documents, the method comprising the steps of:

receiving data that indicates one or more document property values and criteria to be used to determine which of the plurality of documents should be associated with each of the one or more document property values,

wherein the document property values do not appear in a content of the electronic documents;

storing a list of words that are within one document of the plurality of documents; and

storing, in the index, information associating each of the one or more document property values with the one document when the one document satisfies the criteria associated with the document property values.

9. A method of selecting electronic documents from among a plurality of electronic documents, the method comprising the steps of:

storing a document property value in an index in association with information identifying an electronic document, in which the document property value comprises data that does not appear in a content of the electronic document;

receiving a search query;

modifying the search query to create a modified search query by adding to the search query a search term that references the document property value; and

creating a set of search results by searching the index based on the modified search query.

10. A computer-readable medium carrying instructions for selecting electronic documents from among a plurality of electronic documents, the computer-readable medium comprising instructions for performing the steps of:

storing a tag word in an index in association with information identifying an electronic document, in which the tag word comprises data that does not appear in a content of the electronic document;

receiving a search query;

modifying, the search query to create a modified search query by adding to the search query a search term that references the tag word; and

creating a set of search results by searching the index based on the modified search query;

wherein the step of storing includes the steps of:

receiving data that indicates one or more tag words and criteria to be used to determine which of the plurality of documents should be associated with each of the one or more tag words, and in which at least a portion of the data is expressed in a wildcard format;

retrieving a location identifier of each of the documents that are indexed in the index;

matching each location identifier to each of the criteria; and

when one location identifier matches one of the criteria, storing, in the index, information associating such location identifier with one or more of the tag words.

11. A computer-readable medium carrying instructions for selecting electronic documents from among a plurality of electronic documents, the computer-readable medium comprising instructions for performing the steps of:

storing a tag word in an index in association with information identifying an electronic document, in which the tag word comprises data that does not appear in a content of the electronic document;

receiving a search query;

modifying the search query to create a modified search query by adding to the search query a search term that references the tag word; and

creating a set of search results by searching the index based on the modified search query;

wherein the step of storing includes the steps of

receiving specifications of one or more of the documents that are indexed in the index, in which each of the specifications is associated with one or more tag words, and in which one of the specifications is expressed in a wildcard format;

retrieving a location identifier of each of the documents that are indexed in the index;

matching each location identifier to each of the specifications by interpreting the one of the specifications that is in the wildcard format according to one or more wildcard format rules; and

when one location identifier matches one of the specifications, storing, in the index, information associating such location identifier with one or more of the tag words.

12. A computer-readable medium carrying instructions for processing queries that select an electronic document from among a plurality of documents, the computer-readable medium carrying instructions for performing the steps of:

storing a tag word in an index in association with information identifying the electronic document, in which the tag word indicates that access to the electronic document is restricted;

receiving a search query that requests the electronic document;

modifying the search query to create a modified search query by adding a search term that references the tag word; and

creating a set of search results by searching the index based on the modified search query;

wherein the step of storing further includes the steps of:

receiving specifications of one or more of the documents that are indexed in the index, in which each of the specifications is associated with the tag word, and in which each of the specifications is expressed in a wildcard format;

retrieving a location identifier of each of the documents that are indexed in the index;

matching each location identifier to each of the specifications; and

when one location identifier matches one of the specifications, storing, in the index, information associating such location identifier with the tag word.

13. A computer-readable medium carrying instructions for processing queries that select an electronic document from among a plurality of documents, the computer-readable medium comprising instructions for performing the steps of:

storing a tag word in an index in association with information identifying the electronic document, in which the tag word indicates that access to the electronic document is restricted;

receiving a search query that requests the electronic document;

modifying the search query to create a modified search query by adding a search term that references the tag word; and

creating a set of search results by searching the index based on the modified search query;

wherein the step of storing includes the steps of:

receiving specifications of one or more of the documents that are indexed in the index, in which each of the specifications is associated with the tag word, and in which one of the specifications is expressed in a wildcard format;

retrieving a location identifier of each of the documents that are indexed in the index;

matching each location identifier to each of the specifications by interpreting the one of the specifications that is in the wildcard format according to one or more wildcard format rules; and

when one location identifier matches one of the specifications, storing, in the index, information associating such location identifier with the tag words.

14. A computer-readable medium carrying instruction for processing queries that select an electronic document from among a plurality of documents, the computer-readable medium comprising instructions for performing the steps of:

storing a tag word in an index in association with information identifying the electronic document, in which the tag word indicates that access to the electronic document is restricted;

receiving a search query that requests the electronic document;

modifying the search query to create a modified search query by adding a search term that references the tag word; and

creating a set of search results by searching the index based on the modified search query;

wherein the step of storing includes the steps of:

receiving data that indicates one or more tag words and criteria to be used to determine which of the plurality of documents should be associated with each of the one or more tag words, and in which at least a portion of the data is expressed in a wildcard format;

retrieving a location identifier of each of the documents that are indexed in the index;

matching each location identifier to each of the criteria; and

when one location identifier matches one of the criteria, storing, in the index, information associating such location identifier with one or more of the tag words.

15. A computer-readable medium carrying instructions for constructing an index of a plurality of electronic documents for use in selecting electronic documents from among the plurality of electronic documents, the computer-readable medium comprising instructions for performing the steps of:

receiving data that indicates one or more tag words and criteria to be used to determine which of the plurality of documents should be associated with each of the one or more tag words, wherein the tag words do not appear in a content of the electronic documents;

storing a list of words that are within one document of the plurality of documents;

storing, in the index, information associating each of the one or more tag words with the one document when the one document satisfies the criteria associated with the tag words;

wherein the step of receiving data includes the steps of receiving data that indicates one or more tag words and criteria to be used to determine which of the plurality of documents should be associated with each of the one or more tag words, and in which at least a portion of the data is expressed in a wildcard format;

and wherein the step of storing information comprises the steps of retrieving a location identifier of each of the documents; matching each location identifier to each of the criteria; and when one location identifier matches one of the criteria, storing, in the index, information associating such location identifier with one or more of the tag words.

16. A computer-readable medium carrying instructions for constructing an index of a plurality of electronic documents for use in selecting electronic documents from among the plurality of electronic documents, the computer-readable medium carrying instructions for performing the steps of:

receiving data that indicates one or more tag words and criteria to be used to determine which of the plurality of documents should be associated with each of the one or more tag words, wherein the tag words do not appear in a content of the electronic documents;

storing a list of words that are within one document of the plurality of documents;

storing, in the index, information associating each of the one or more tag words with the one document when the one document satisfies the criteria associated with the tag words;

wherein the step of receiving data includes the steps of receiving specifications of one or more of the documents that are indexed in the index, in which each of the specifications is associated with one or more tag words, and in which one of the specifications is expressed in a wildcard format;

and wherein the step of storing information comprises the steps of:

retrieving a location identifier of each of the documents that are indexed in the index;

matching each location identifier to each of the specifications by interpreting the one of the specifications that is in the wildcard format according to one or more wildcard format rules; and

when one location identifier matches one of the specifications, storing, in the index, information associating such location identifier with one or more of the tag words.

17. A computer-readable medium carrying instructions for constructing an index of a plurality of electronic documents for use in selecting electronic document from among the plurality of electronic documents, the computer-readable medium comprising instructions for performing the steps of:

receiving data that indicates one or more document property values and criteria to be used to determine which of the plurality of documents should be associated with each of the one or more document property values, wherein the document property values do not appear in a content of the electronic documents;

storing a list of words that are within one document of the plurality of documents; and

storing, in the index, information associating each of the one or more document property values with the one document when the one document satisfies the criteria associated with the document property values.

18. A computer-readable medium carrying instructions for selecting electronic documents from among a plurality of electronic documents, the computer-readable medium comprising instructions for performing the steps of:

storing a document property value in an index in association with information identifying an electronic document, in which the document property value comprises data that does not appear in a content of the electronic document;

receiving a search query;

modifying the search query to create a modified search query by adding to the search query a search term that references the document property value; and

creating a set of search results by searching the index based on the modified search query.

* * * * *

US006356905B1

(12) **United States Patent**
**Gershman et al.**

(10) **Patent No.: US 6,356,905 B1**
(45) **Date of Patent: Mar. 12, 2002**

(54) **SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR MOBILE COMMUNICATION UTILIZING AN INTERFACE SUPPORT FRAMEWORK**

(75) Inventors: **Anatole Vitaly Gershman**, Chicago; **Kishore Sundaram Swaminathan**, Downers Grove; **James L. Meyers**, Chicago; **Andrew Ernest Fano**, Evanston, all of IL (US)

(73) Assignee: **Accenture LLP**, Palo Alto, CA (US)

( * ) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/263,252**

(22) Filed: **Mar. 5, 1999**

(51) **Int. Cl.**[7] ..... **G06F 17/30**

(52) **U.S. Cl.** ..... **707/10**; 707/3; 707/5; 707/102; 705/26; 705/35; 709/203; 709/219

(58) **Field of Search** ..... 707/3, 4, 10, 102, 707/5; 235/462, 472; 705/26, 35; 709/203, 219

(56) **References Cited**

U.S. PATENT DOCUMENTS

| | | | |
|---|---|---|---|
| 5,279,882 A | 1/1994 | Daude et al. | 428/192 |
| 5,519,608 A | 5/1996 | Kupiec | 364/419.08 |
| 5,606,602 A | 2/1997 | Johnson et al. | 379/115 |
| 5,640,193 A | 6/1997 | Wellner | 348/7 |
| 5,673,322 A | 9/1997 | Pepe et al. | 389/49 |

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

| | | |
|---|---|---|
| EP | 0 853 287 A2 | 7/1998 |
| EP | 0883313 A2 | 12/1998 |
| EP | 0890907 A1 | 1/1999 |
| WO | WO 97/17815 | 5/1997 |
| WO | WO/97/40451 | 10/1997 |
| WO | WO 97/45814 | 12/1997 |
| WO | WO 98/03923 | 1/1998 |
| WO | WO 98/06055 | 2/1998 |
| WO | WO 98/10541 | 3/1998 |
| WO | WO 98/11744 | 3/1998 |
| WO | WO 98/12833 | 3/1998 |
| WO | WO 98/24036 | 6/1998 |
| WO | WO 98/24050 | 6/1998 |
| WO | 98/35469 | 8/1998 |
| WO | WO 98/39909 | 9/1998 |
| WO | WO 98/40823 | 9/1998 |
| WO | WO 98/49813 | 11/1998 |
| WO | WO 98/52371 | 11/1998 |
| WO | 98/47295 | 12/1998 |
| WO | WO 98/57474 | 12/1998 |
| WO | WO 98/58476 | 12/1998 |
| WO | WO 99/01969 | 1/1999 |

OTHER PUBLICATIONS

D. Brugali, "Techniques to build and reuse assets", Practical Software Reuse, ESSI–Surprise 1998, pp. 7–1 to 7–11.

D. Brugali et al., "Agent technology: a new frontier for the development of application frameworks?", Object–Oriented Application Frameworks, Wiley 1998, 9 pp.

Falchuk et al., "AgenSys: A Mobile Agent System for Digital Media Access and Ineraction on an Internet", Univ. of Ottawa, Dept. of El. and Comp. Eng., 1997, pp. 1876–1880.

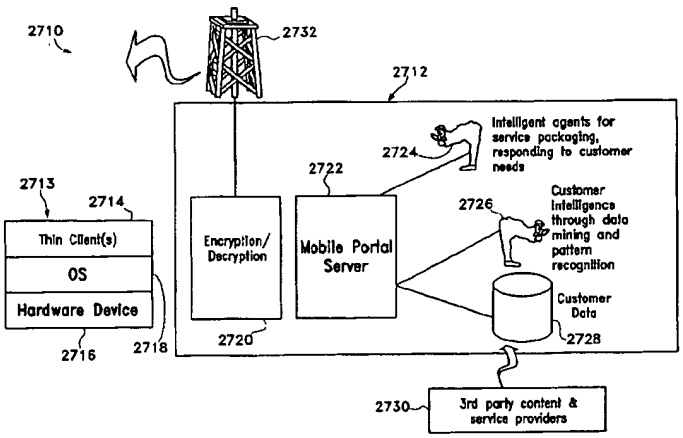
(List continued on next page.)

*Primary Examiner*—Jean R. Homere
(74) *Attorney, Agent, or Firm*—Morrison & Foerster LLP

(57) **ABSTRACT**

A system is disclosed that facilitates web-based information retrieval and display system. A wireless phone or similar hand-held wireless device with Internet Protocol capability is combined with other peripherals to provide a portable portal into the Internet. The wireless device prompts a user to input information of interest to the user. This information is transmitted a query to a service routine (running on a Web server). The service routine then queries the Web to find price, shipping and availability information from various Web suppliers. This information is then available for use by various applications through an interface support framework.

**18 Claims, 29 Drawing Sheets**



U.S. PATENT DOCUMENTS

| | | | | | |
|---|---|---|---|---|---|
| 5,732,074 | A | | 3/1998 | Spaur et al. | 370/313 |
| 5,850,442 | A | | 12/1998 | Muftic | 380/21 |
| 5,854,624 | A | | 12/1998 | Grant | 345/169 |
| 5,913,210 | A | * | 6/1999 | Call | 707/4 |
| 5,937,163 | A | * | 8/1999 | Lee et al. | 395/200.48 |
| 5,938,727 | A | * | 8/1999 | Ikeda | 709/218 |
| 5,950,173 | A | * | 9/1999 | Perkowski | 705/26 |
| 5,971,277 | A | * | 10/1999 | Cragun et al. | 235/462.01 |
| 5,978,773 | A | * | 11/1999 | Hudetz et al. | 705/23 |
| 5,979,757 | A | * | 11/1999 | Tracy et al. | 235/383 |
| 5,992,752 | A | * | 11/1999 | Wilz, Sr. et al. | 235/472.01 |
| 6,134,548 | A | * | 10/2000 | Gottsman et al. | 707/5 |
| 6,202,062 | B1 | * | 3/2001 | Cameron et al. | 707/3 |

OTHER PUBLICATIONS

U. S. patent application Ser. No. 09/263,969, filed Mar. 5, 1999, "Mobile Communication and Computing System and Method".

U. S. patent application Ser. No. 09/263,927, filed Mar. 5, 1999, "Mobile Communication System and Method for A Shopper Agent".

U. S. patent application Ser. No. 09/263,926, filed Mar. 5, 1999, "A System, Method and Article of Manufacture for Advanced Mobile Health Care Processing".

U. S. patent application Ser. No. 09/263,920, filed Mar. 5, 1999, "Dynamic Configuration System and Method for A Mobile Communication Network".

U. S. patent application Ser. No. 09/263,251, filed Mar. 5, 1999, "A System for Utilizing A Transaction Interface in A Mobile Communication Network".

U. S. patent application Ser. No. 09/272,828, filed Mar. 19, 1999, "System and Method for Inputting, Retrieving, Organizing and Analyzing Data".

Bob Emmerson; The Mobile Intranet: The next generation of GSM services will offer faster data rates and smarter messaging; May 1998; BYTE Magazine, pp. 1–7.

Timo Alanko, Markku Kojo, Mika Liljeberg, Kimmo Raatikainen; Mobile access to the Internet: a mediator-based solution; Internet Research: Electronic Networking Applications and Policy vol. 9, No. 1, pp. 58–65, 1999.

Andrezej Duda, Stephane Perret; A Network Programming Model for Mobile Applications nd Information Access; Proceedings JENC7, pp. 141.1–149.9, DATE 7.

Chu-Sing Yang, Kun-da Wu, Chun-Wei Tseng; Support an Efficient Connection for Mobile IP; Proceedings, Ninth International Workshop on Database and Expert Systems Applications; Aug. 1998, IEEE, Computer Society, pp. 514–519.

Mary Cameron Cupito; Emerging technologies: Has Their time come? Enterprise Integration; Health Management Technology; Dec. 1998, pp. 12–16.

Toh Han Shih; Online life-line; Wired for Business; Singapore Business Times; Dec. 1998, pp. ?.

Chris Bradley; Remote and Mobile Computing With TCP/IP; Enterprise Systems Journal; pp. 38–48.

Enhanced Services: Telecom customers will soon have one-stop, easy-to-use access to their services portfolio form anywhere, at any time, and in any way; EDGE, on & about AT&T; May 1997, pp. 1–2, Authors?.

Nokia, Ericsson, Unwired Planet and Motorola unite to create an open common protocol for interactive wireless applications; Jun. 26, 1997, pp. 1–3.

Unisource in GSM trial of mobile electronic banking and shopping; Mobile Communications; Mar. 20, 1997, pp. 1–3, Authors?.

Dynamic Mobile Data Announces Mobile Server Wireless Solution For Enterprise and internet Access; Mar. 1999, pp. 1–2, Authors?.

Philip R. Cohen, Adam Cheyer, Michelle Wang, Soon Cheol Baeg; An Open Agent Architecture; Software Agent Papers, AAAI Spring Symposium 1994, pp. 1–129.

Katia Sycara, Anandeep S. Pannu; The RETSINA Multiagent System: Towards Integrating Planning, Execution and Information Gathering; Proceedings of the Second International Conference on Autonomous Agents, May 1998, pp. 350–351.

* cited by examiner

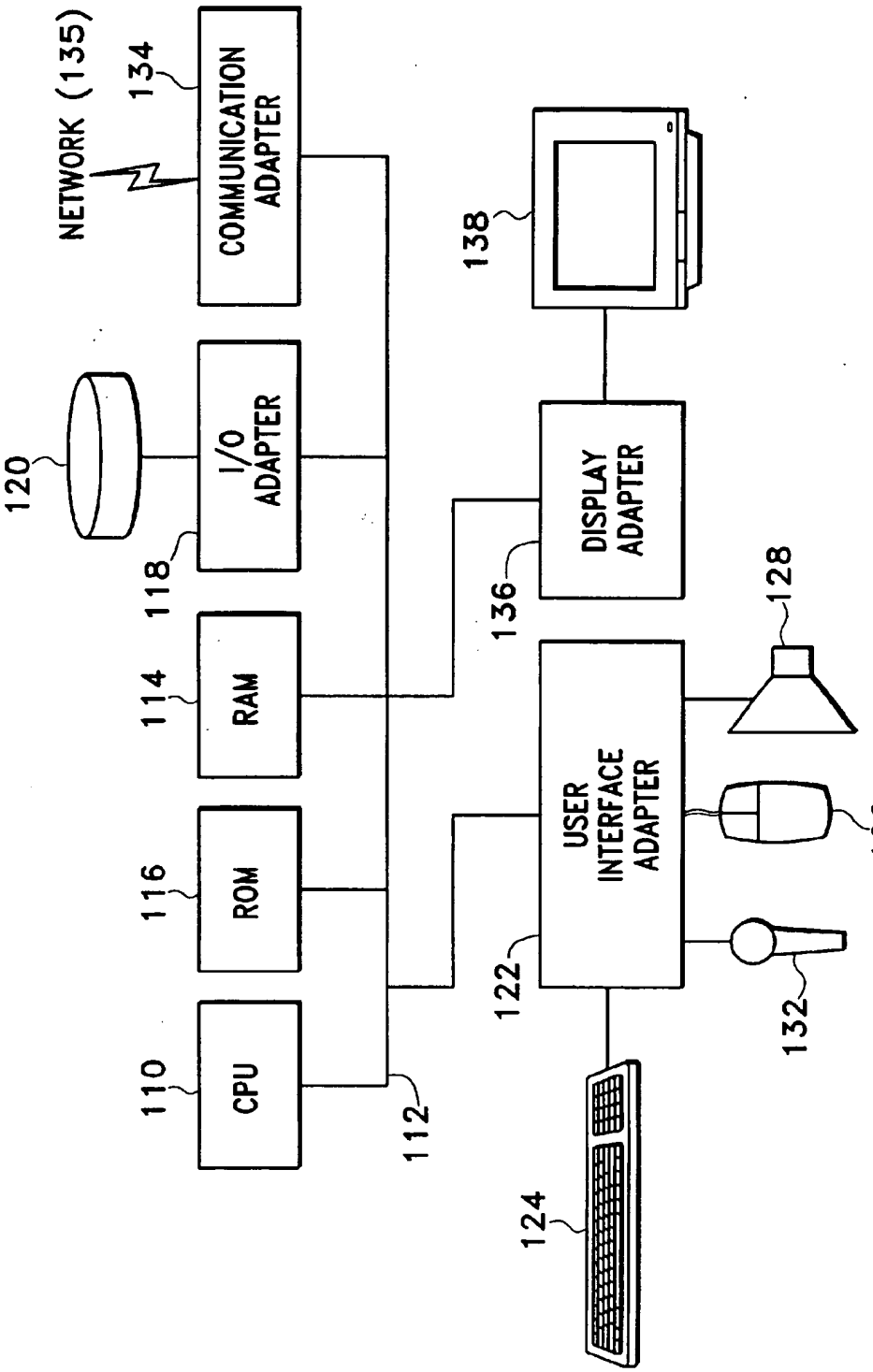

NETWORK (135)
134
COMMUNICATION ADAPTER
138
120
I/O ADAPTER
DISPLAY ADAPTER
118
136
128
114
RAM
USER INTERFACE ADAPTER
116
ROM
126
122
132
110
CPU
112
124

Fig. 1

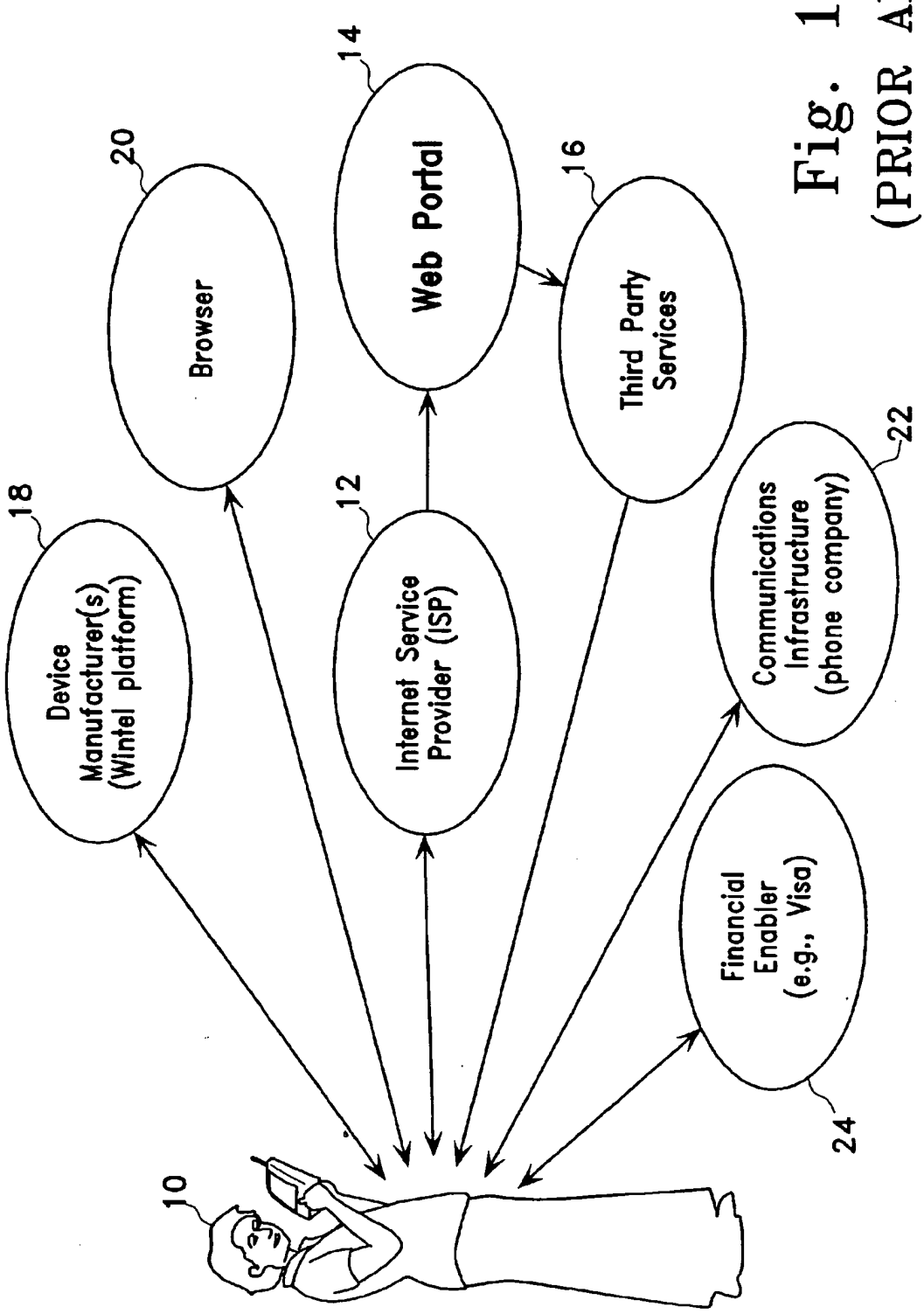



Fig. 1A
(PRIOR ART)

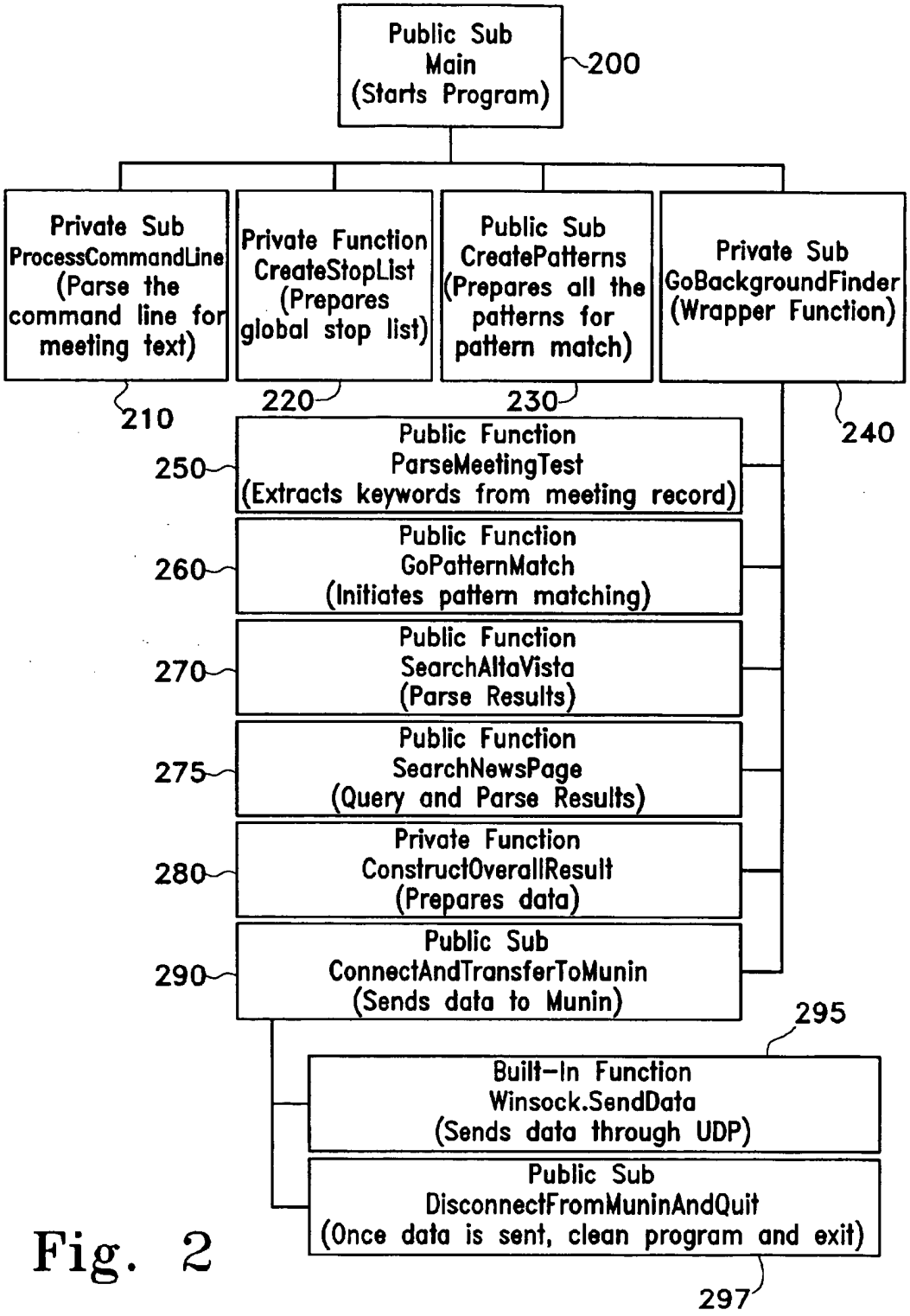

Public Sub
Main
(Starts Program)
200
Private Sub
ProcessCommandLine
(Parse the
command line for
meeting text)
Private Function
CreateStopList
(Prepares
global stop list)
Public Sub
CreatePatterns
(Prepares all the
patterns for
pattern match)
Private Sub
GoBackgroundFinder
(Wrapper Function)
210
220
230
240
250
Public Function
ParseMeetingTest
(Extracts keywords from meeting record)
260
Public Function
GoPatternMatch
(Initiates pattern matching)
270
Public Function
SearchAltaVista
(Parse Results)
275
Public Function
SearchNewsPage
(Query and Parse Results)
280
Private Function
ConstructOverallResult
(Prepares data)
290
Public Sub
ConnectAndTransferToMunin
(Sends data to Munin)
295
Built-In Function
Winsock.SendData
(Sends data through UDP)
Public Sub
DisconnectFromMuninAndQuit
(Once data is sent, clean program and exit)
297

Fig. 2

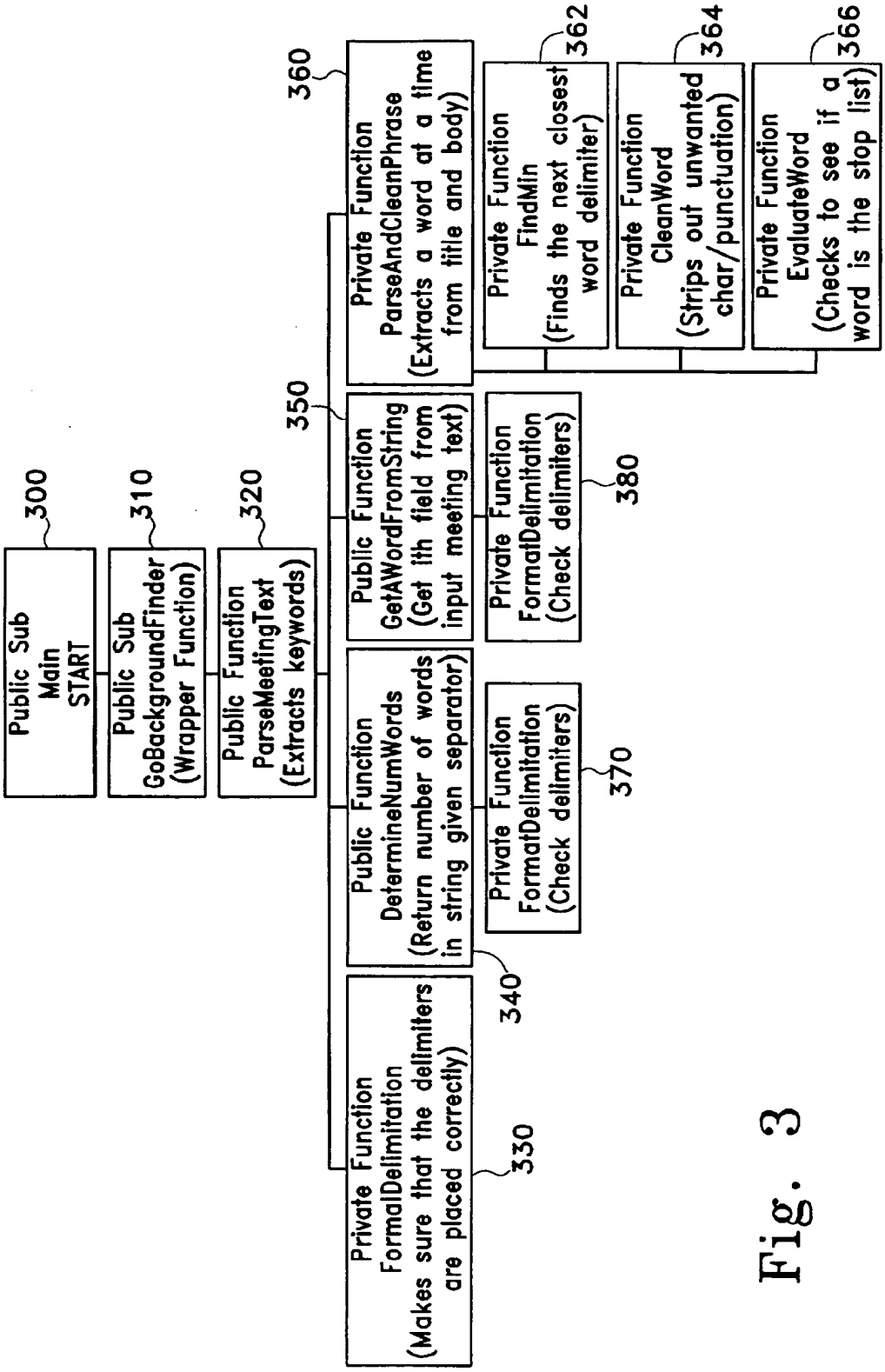

Public Sub Main START
300
Public Sub GoBackgroundFinder (Wrapper Function)
310
Public Function ParseMeetingText (Extracts keywords)
320
Public Function GetAWordFromString (Get ith field from input meeting text)
350
Public Function DetermineNumWords (Return number of words in string given separator)
340
Private Function FormatDelimitation (Makes sure that the delimiters are placed correctly)
330
Private Function FormatDelimitation (Check delimiters)
370
Private Function FormatDelimitation (Check delimiters)
380
Private Function ParseAndCleanPhrase (Extracts a word at a time from title and body)
360
Private Function FindMin (Finds the next closest word delimiter)
362
Private Function CleanWord (Strips out unwanted char/punctuation)
364
Private Function EvaluateWord (Checks to see if a word is the stop list)
366

Fig. 3

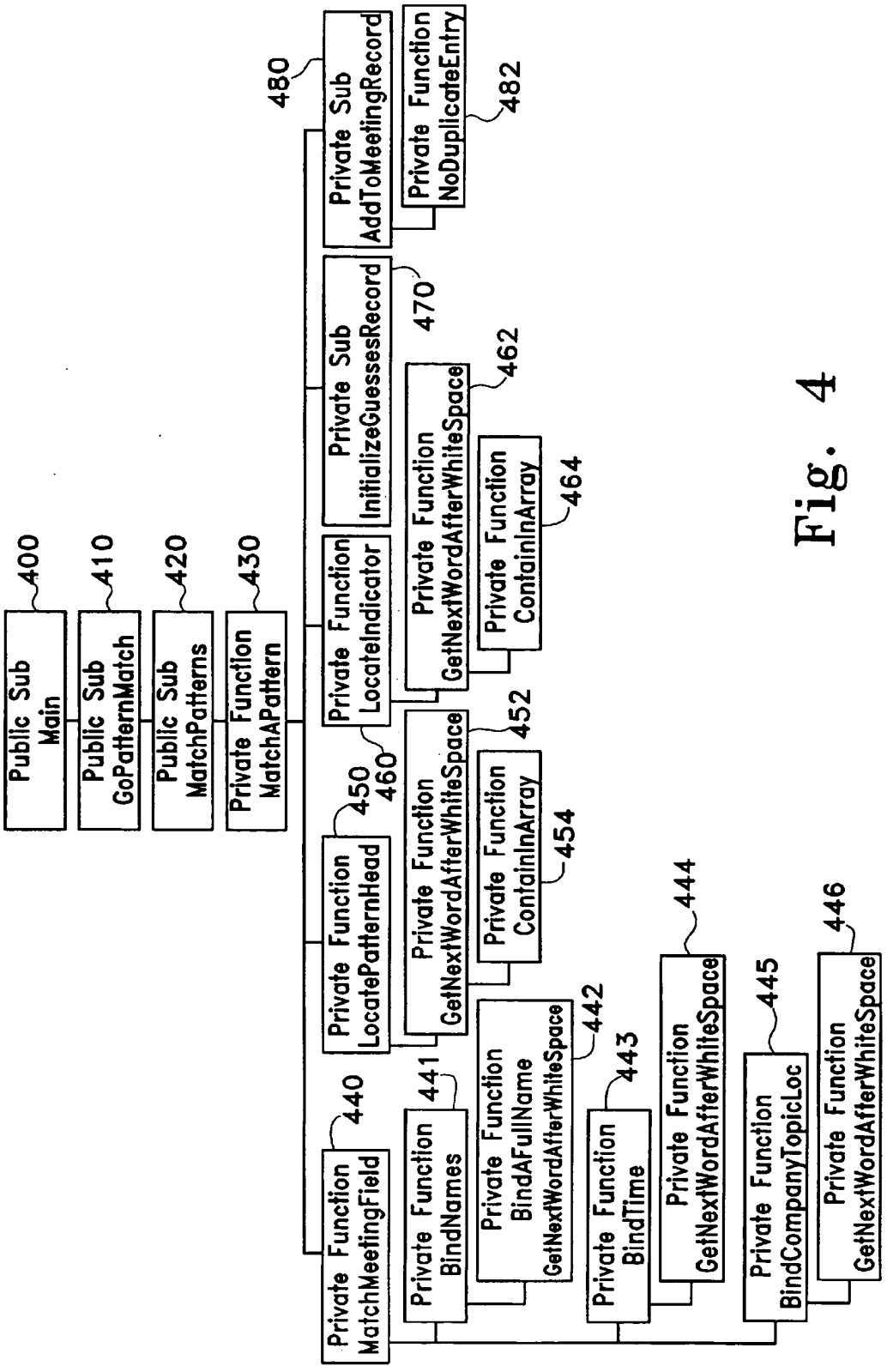

Public Sub Main
400
Public Sub GoPatternMatch
410
Public Sub MatchPatterns
420
Private Function MatchAPattern
430
Private Function MatchMeetingField
440
Private Function BindNames
441
Private Function BindAFullName
442
Private Function BindTime
443
Private Function GetNextWordAfterWhiteSpace
444
Private Function BindCompanyTopicLoc
445
Private Function GetNextWordAfterWhiteSpace
446
Private Function LocatePatternHead
450
Private Function GetNextWordAfterWhiteSpace
452
Private Function ContainInArray
454
Private Function LocateIndicator
460
Private Function GetNextWordAfterWhiteSpace
462
Private Function ContainInArray
464
Private Sub InitializeGuessesRecord
470
Private Sub AddToMeetingRecord
480
Private Function NoDuplicateEntry
482

Fig. 4

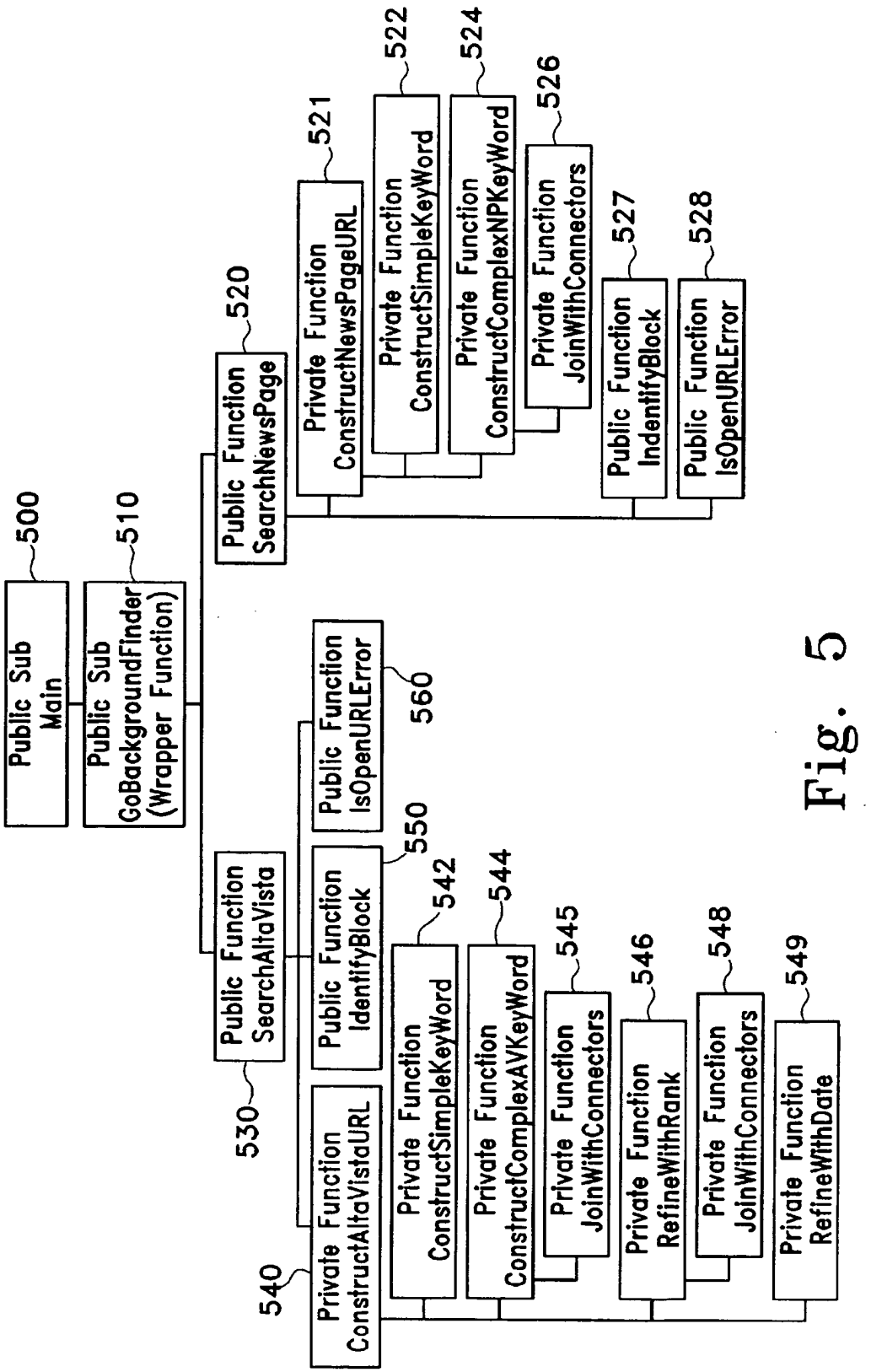

Public Sub Main
500
Public Sub GoBackgroundFinder (Wrapper Function)
510
Public Function SearchNewsPage
520
Private Function ConstructNewsPageURL
521
Private Function ConstructSimpleKeyWord
522
Private Function ConstructComplexNPKeyWord
524
Private Function JoinWithConnectors
526
Public Function IndentifyBlock
527
Public Function IsOpenURLError
528
Public Function SearchAltaVista
530
Private Function ConstructAltaVistaURL
540
Public Function IsOpenURLError
560
Public Function IdentifyBlock
550
Private Function ConstructSimpleKeyWord
542
Private Function ConstructComplexAVKeyWord
544
Private Function JoinWithConnectors
545
Private Function RefineWithRank
546
Private Function JoinWithConnectors
548
Private Function RefineWithDate
549

Fig. 5

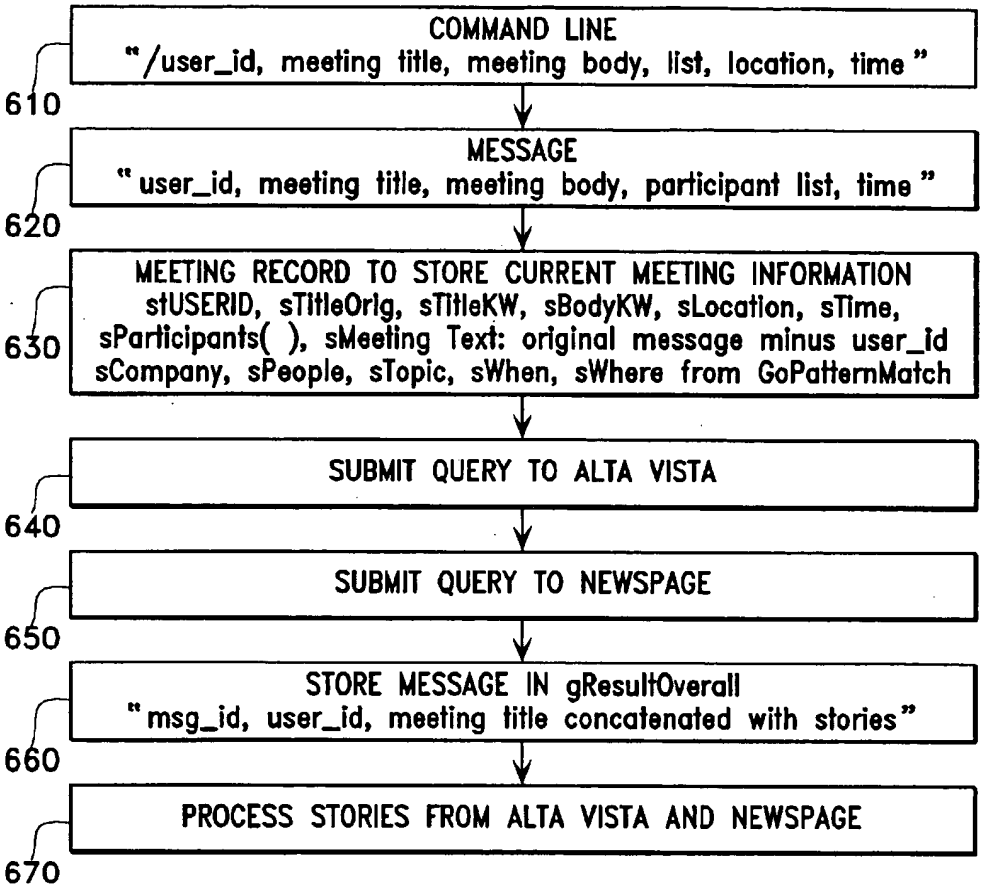

COMMAND LINE
"/user_id, meeting title, meeting body, list, location, time"
610
MESSAGE
"user_id, meeting title, meeting body, participant list, time"
620
MEETING RECORD TO STORE CURRENT MEETING INFORMATION
stUSERID, sTitleOrig, sTitleKW, sBodyKW, sLocation, sTime,
sParticipants( ), sMeeting Text: original message minus user_id
sCompany, sPeople, sTopic, sWhen, sWhere from GoPatternMatch
630
SUBMIT QUERY TO ALTA VISTA
640
SUBMIT QUERY TO NEWSPAGE
650
STORE MESSAGE IN gResultOverall
"msg_id, user_id, meeting title concatenated with stories"
660
PROCESS STORIES FROM ALTA VISTA AND NEWSPAGE
670

Fig. 6

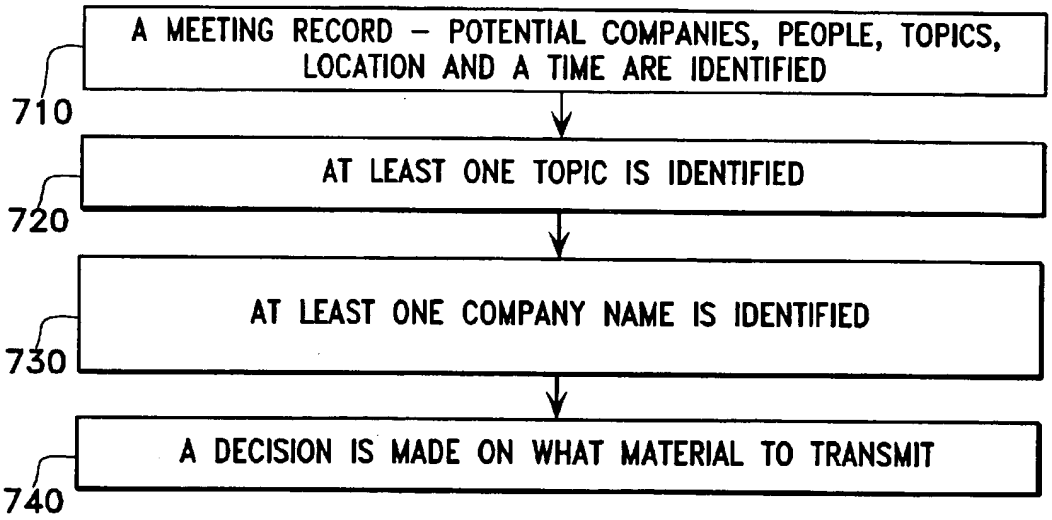

A MEETING RECORD – POTENTIAL COMPANIES, PEOPLE, TOPICS, LOCATION AND A TIME ARE IDENTIFIED
710
AT LEAST ONE TOPIC IS IDENTIFIED
720
AT LEAST ONE COMPANY NAME IS IDENTIFIED
730
A DECISION IS MADE ON WHAT MATERIAL TO TRANSMIT
740

Fig. 7

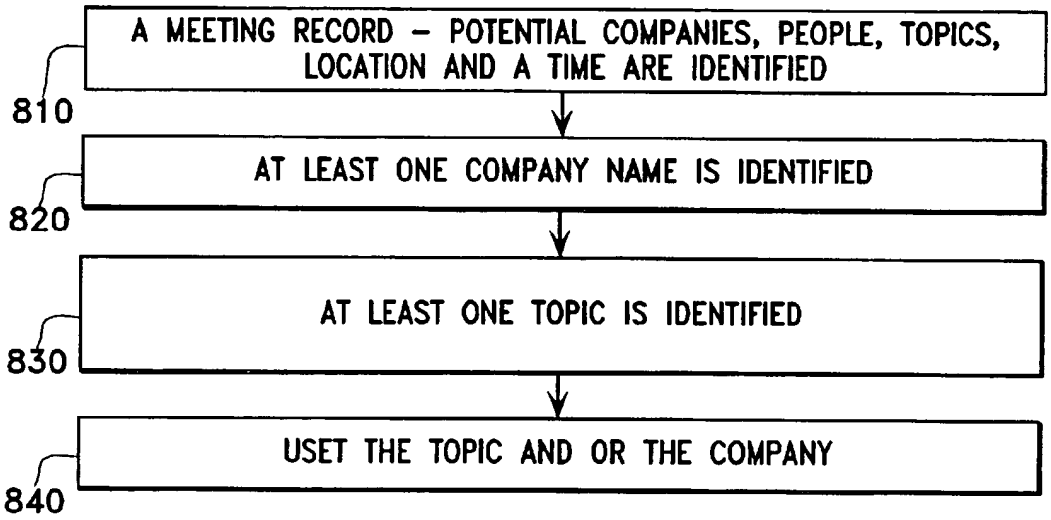

A MEETING RECORD – POTENTIAL COMPANIES, PEOPLE, TOPICS, LOCATION AND A TIME ARE IDENTIFIED
810
AT LEAST ONE COMPANY NAME IS IDENTIFIED
820
AT LEAST ONE TOPIC IS IDENTIFIED
830
USET THE TOPIC AND OR THE COMPANY
840

Fig. 8

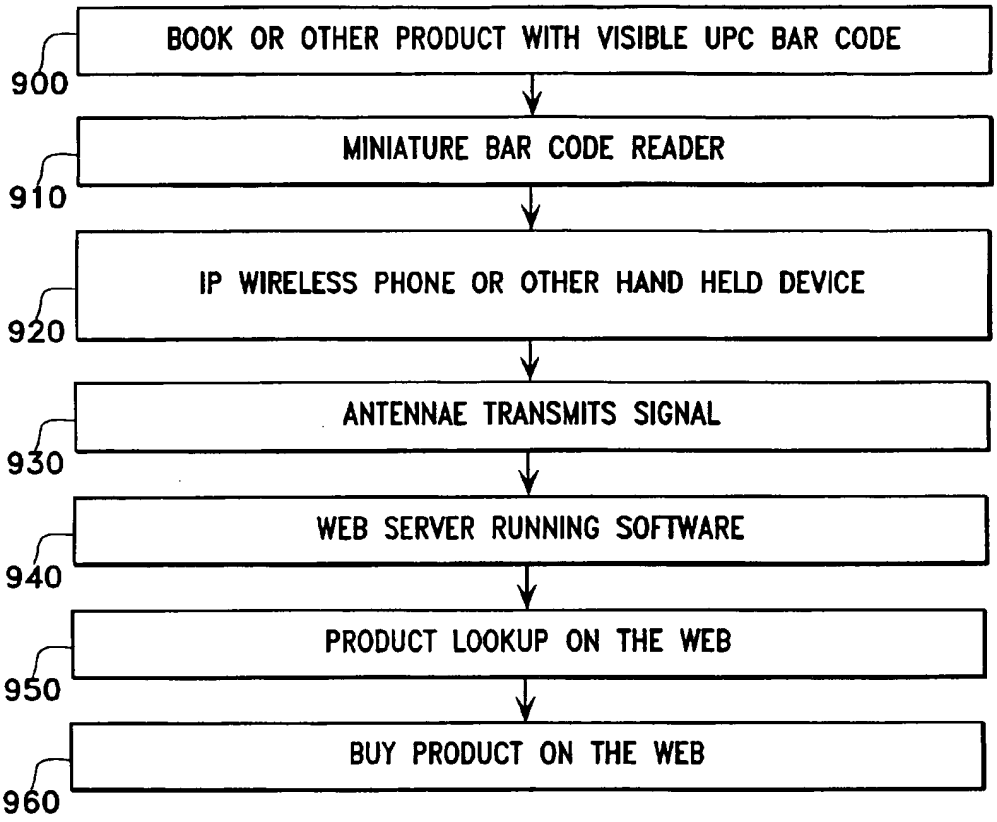

BOOK OR OTHER PRODUCT WITH VISIBLE UPC BAR CODE
900
MINIATURE BAR CODE READER
910
IP WIRELESS PHONE OR OTHER HAND HELD DEVICE
920
ANTENNAE TRANSMITS SIGNAL
930
WEB SERVER RUNNING SOFTWARE
940
PRODUCT LOOKUP ON THE WEB
950
BUY PRODUCT ON THE WEB
960

Fig. 9

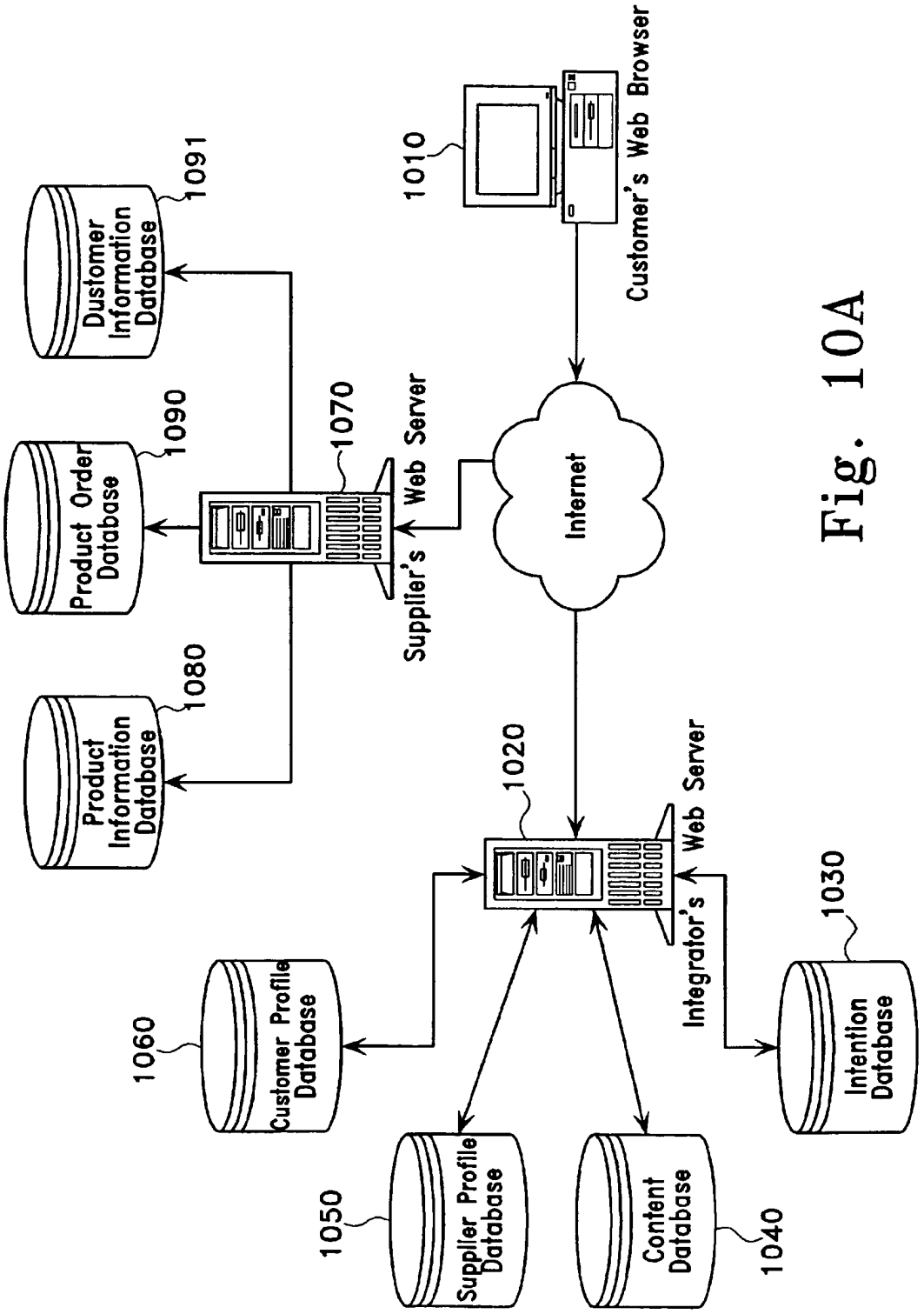

Customer's Web Browser
1010
Internet
Supplier's Web Server
1070
Integrator's Web Server
1020
Customer Information Database
1091
Product Order Database
1090
Product Information Database
1080
Customer Profile Database
1060
Supplier Profile Database
1050
Content Database
1040
Intention Database
1030

Fig. 10A

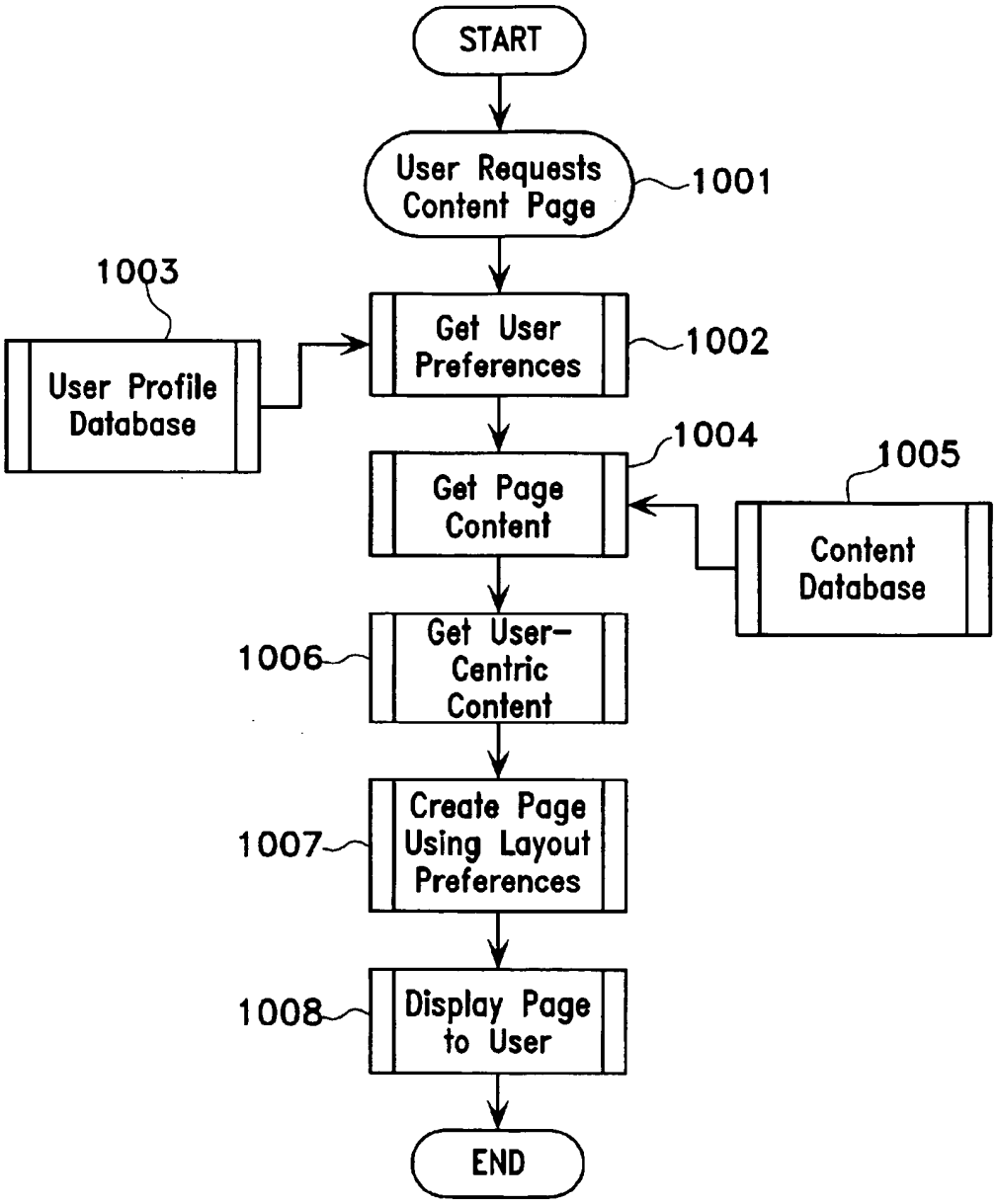

START
User Requests Content Page
1001
1003
User Profile Database
Get User Preferences
1002
1004
Get Page Content
1005
Content Database
1006
Get User-Centric Content
1007
Create Page Using Layout Preferences
1008
Display Page to User
END

Fig. 10B

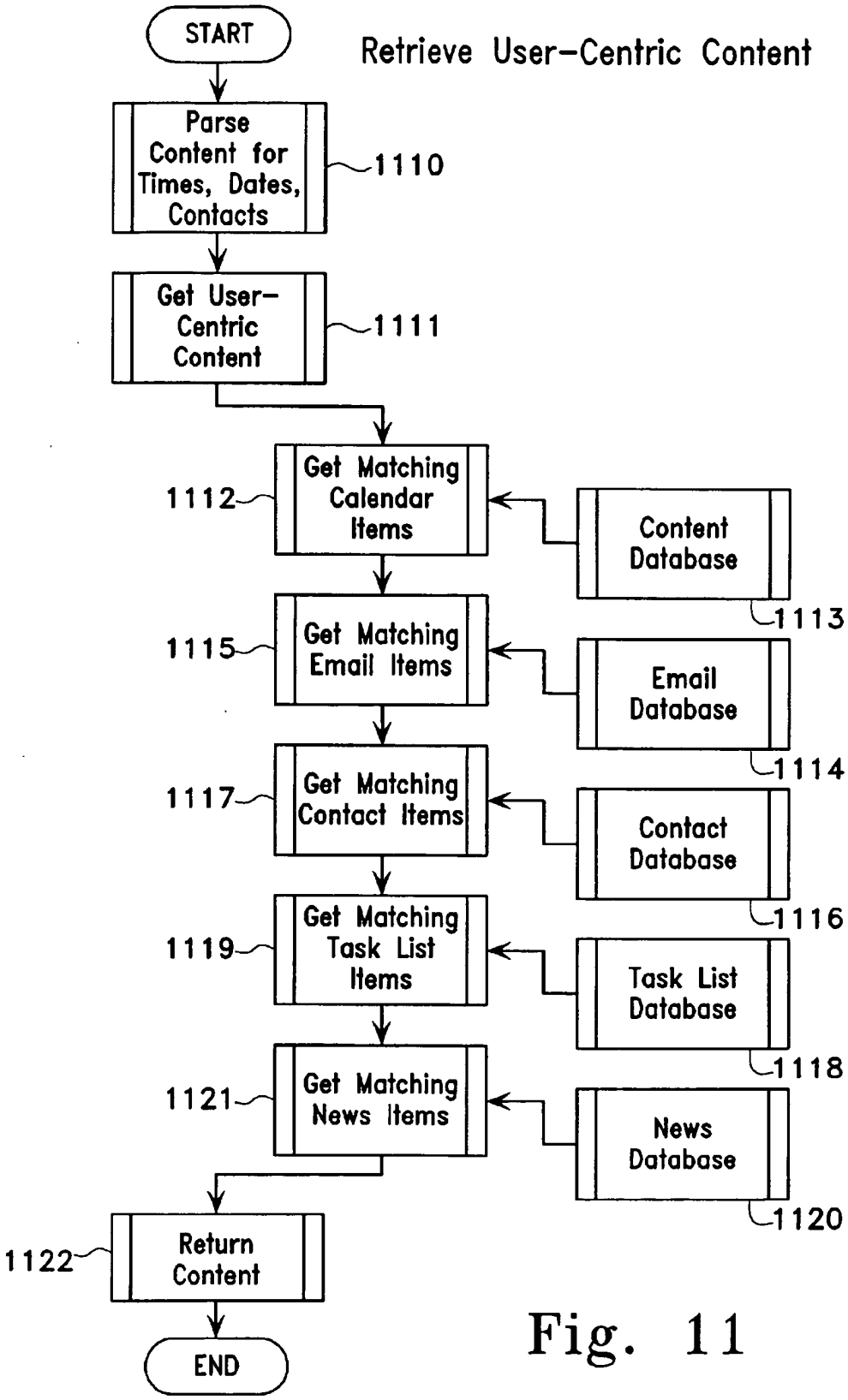

START
Retrieve User-Centric Content
Parse Content for Times, Dates, Contacts
1110
Get User-Centric Content
1111
1112
Get Matching Calendar Items
Content Database
1113
1115
Get Matching Email Items
Email Database
1114
1117
Get Matching Contact Items
Contact Database
1116
1119
Get Matching Task List Items
Task List Database
1118
1121
Get Matching News Items
News Database
1120
1122
Return Content
END

Fig. 11

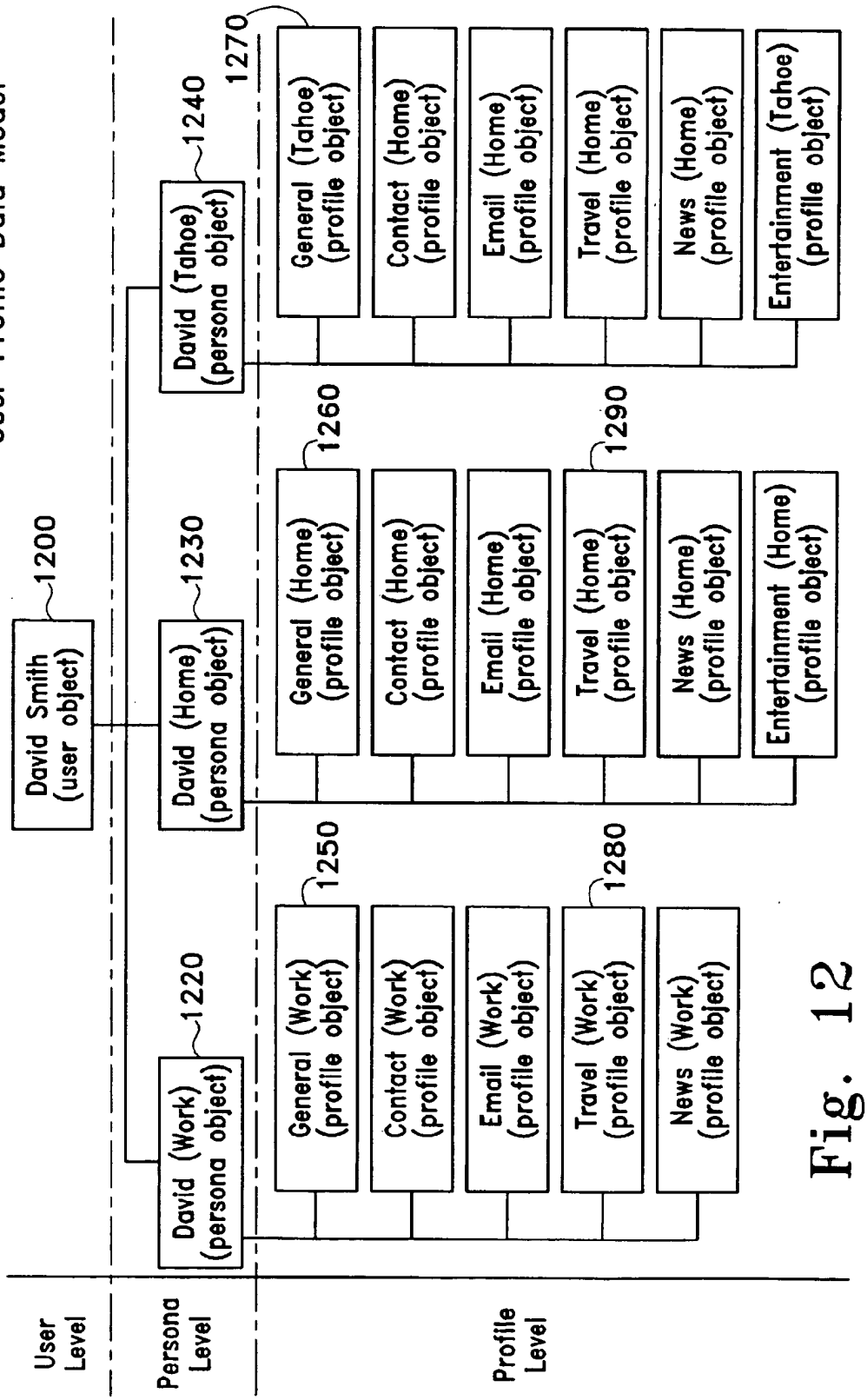

User Profile Data Model
David Smith (user object)
1200
David (Work) (persona object)
1220
David (Home) (persona object)
1230
David (Tahoe) (persona object)
1240
General (Work) (profile object)
Contact (Work) (profile object)
Email (Work) (profile object)
Travel (Work) (profile object)
News (Work) (profile object)
1250
1280
General (Home) (profile object)
Contact (Home) (profile object)
Email (Home) (profile object)
Travel (Home) (profile object)
News (Home) (profile object)
Entertainment (Home) (profile object)
1260
1290
General (Tahoe) (profile object)
Contact (Home) (profile object)
Email (Home) (profile object)
Travel (Home) (profile object)
News (Home) (profile object)
Entertainment (Tahoe) (profile object)
1270
User Level
Persona Level
Profile Level

Fig. 12

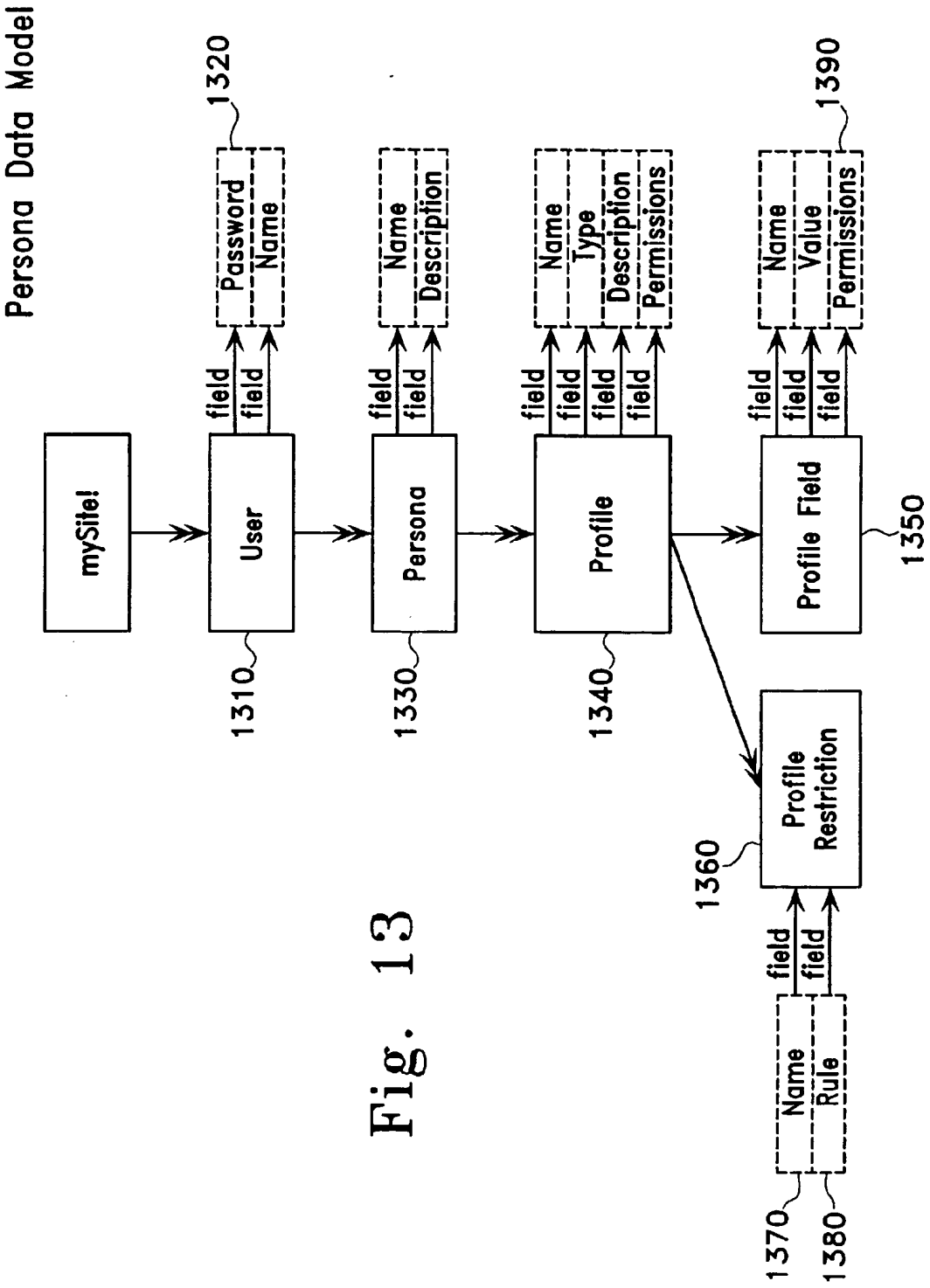

Persona Data Model
mySite!
User
Persona
Profile
Profile Field
Profile Restriction
field
Password
Name
Description
Type
Permissions
Value
Rule
1310
1320
1330
1340
1350
1360
1370
1380
1390

Fig. 13

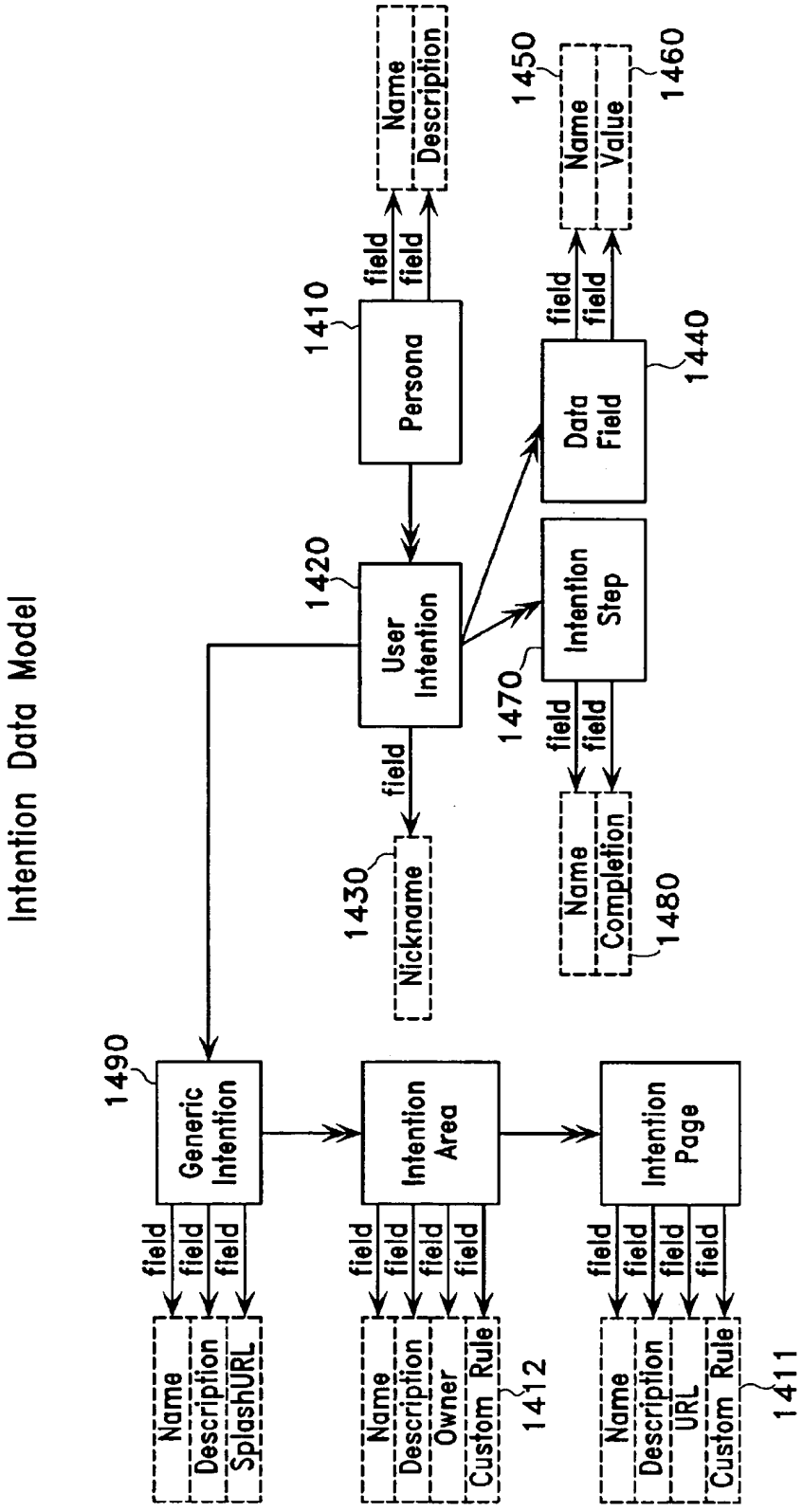

Intention Data Model
Persona
1410
Name
Description
field
field
User Intention
1420
Nickname
1430
field
Data Field
1440
1450
Name
Value
1460
Intention Step
1470
Name Completion
1480
Generic Intention
1490
Name
Description
SplashURL
Intention Area
Name
Description
Owner
Custom Rule
1412
Intention Page
Name
Description
URL
Custom Rule
1411

Fig. 14

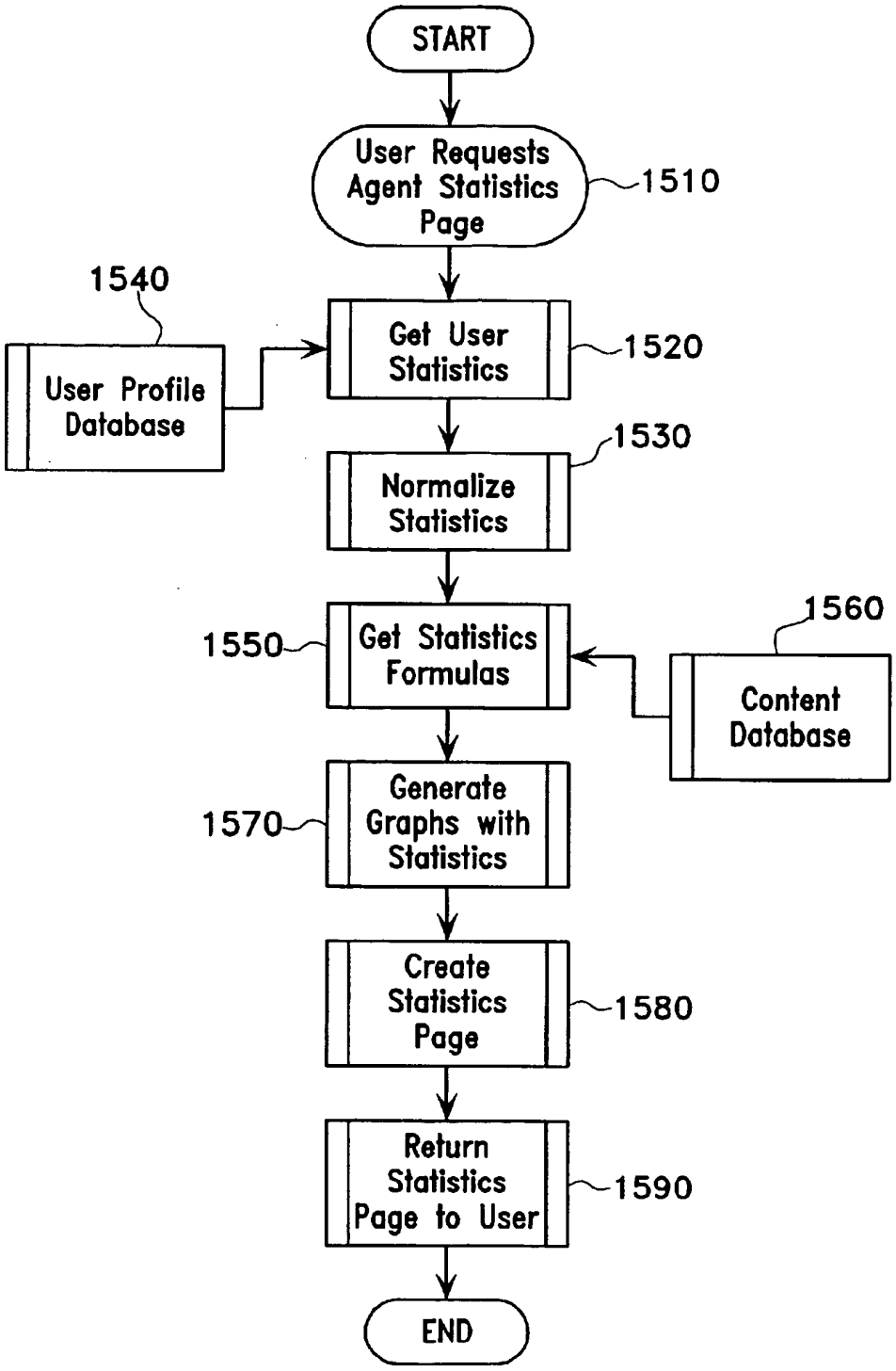

START
User Requests Agent Statistics Page
1510
1540
User Profile Database
Get User Statistics
1520
1530
Normalize Statistics
1560
1550
Get Statistics Formulas
Content Database
Generate Graphs with Statistics
1570
Create Statistics Page
1580
Return Statistics Page to User
1590
END

Fig. 15

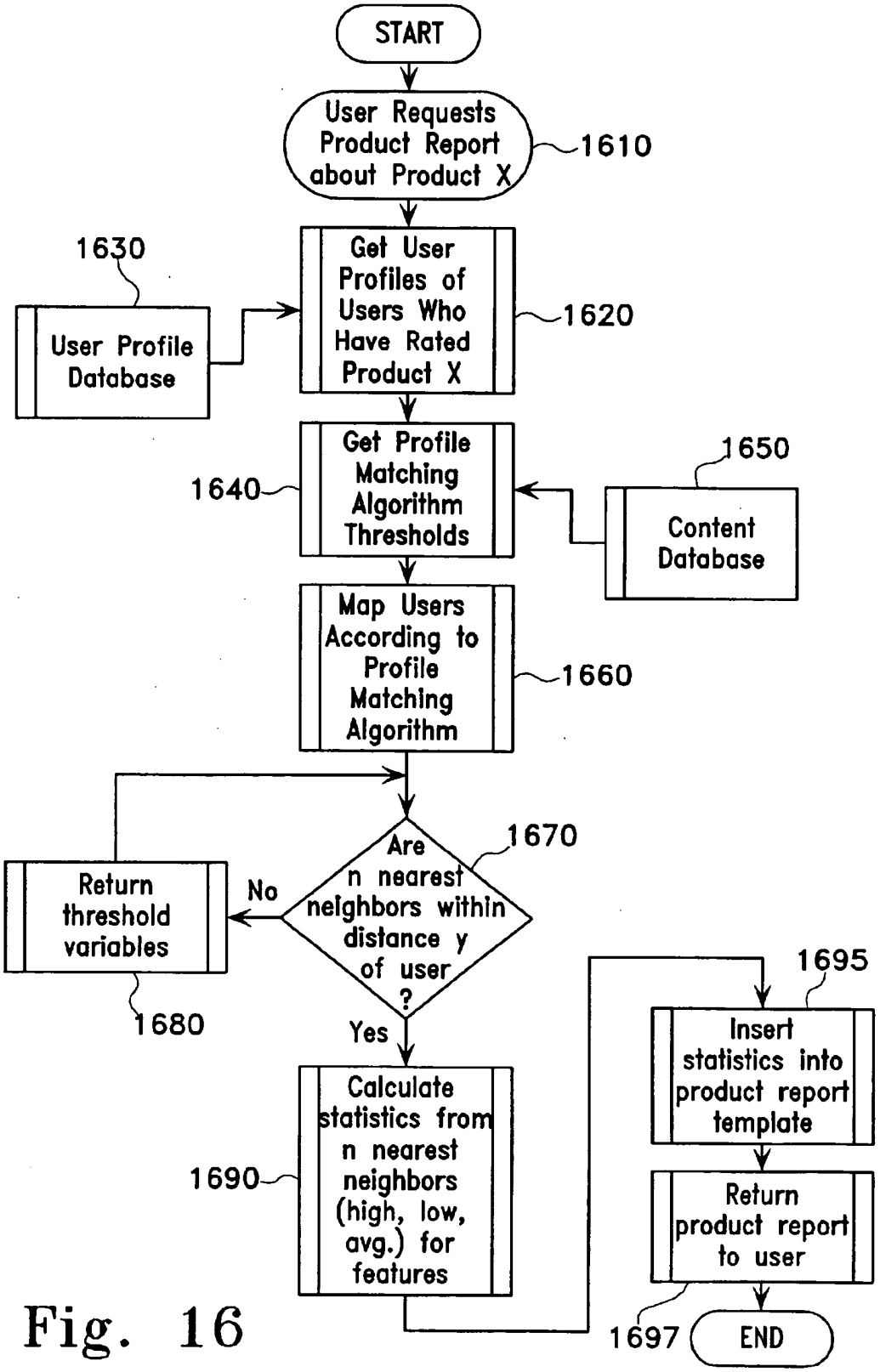

START
User Requests Product Report about Product X
1610
1630
User Profile Database
Get User Profiles of Users Who Have Rated Product X
1620
1640
Get Profile Matching Algorithm Thresholds
1650
Content Database
Map Users According to Profile Matching Algorithm
1660
1670
Are n nearest neighbors within distance y of user ?
No
Return threshold variables
1680
Yes
Calculate statistics from n nearest neighbors (high, low, avg.) for features
1690
1695
Insert statistics into product report template
Return product report to user
1697
END

Fig. 16

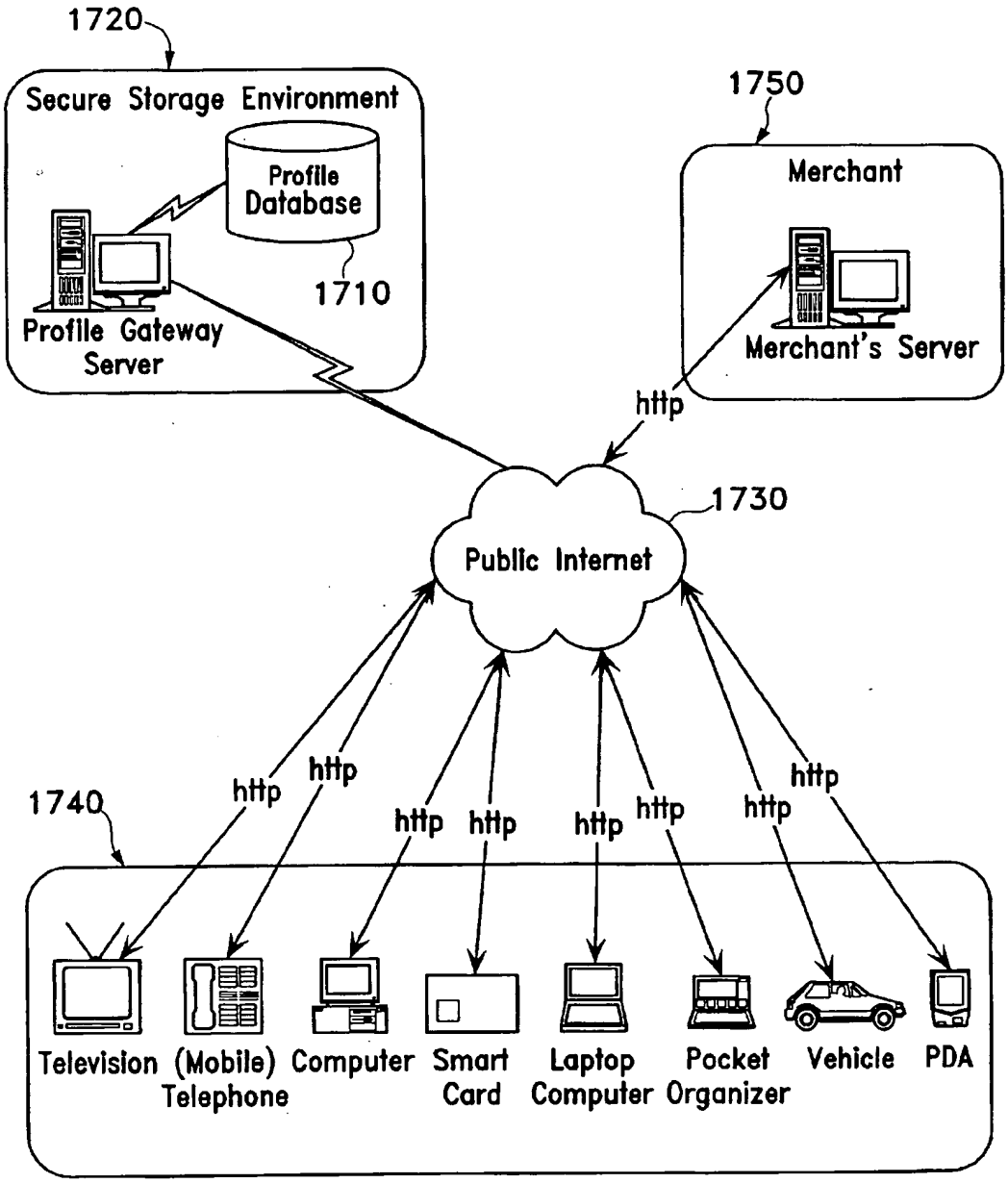

1720
Secure Storage Environment
Profile Database
1710
Profile Gateway Server
1750
Merchant
Merchant's Server
http
1730
Public Internet
1740
http
http
http
http
http
http
http
http
Television
(Mobile) Telephone
Computer
Smart Card
Laptop Computer
Pocket Organizer
Vehicle
PDA

Fig. 17

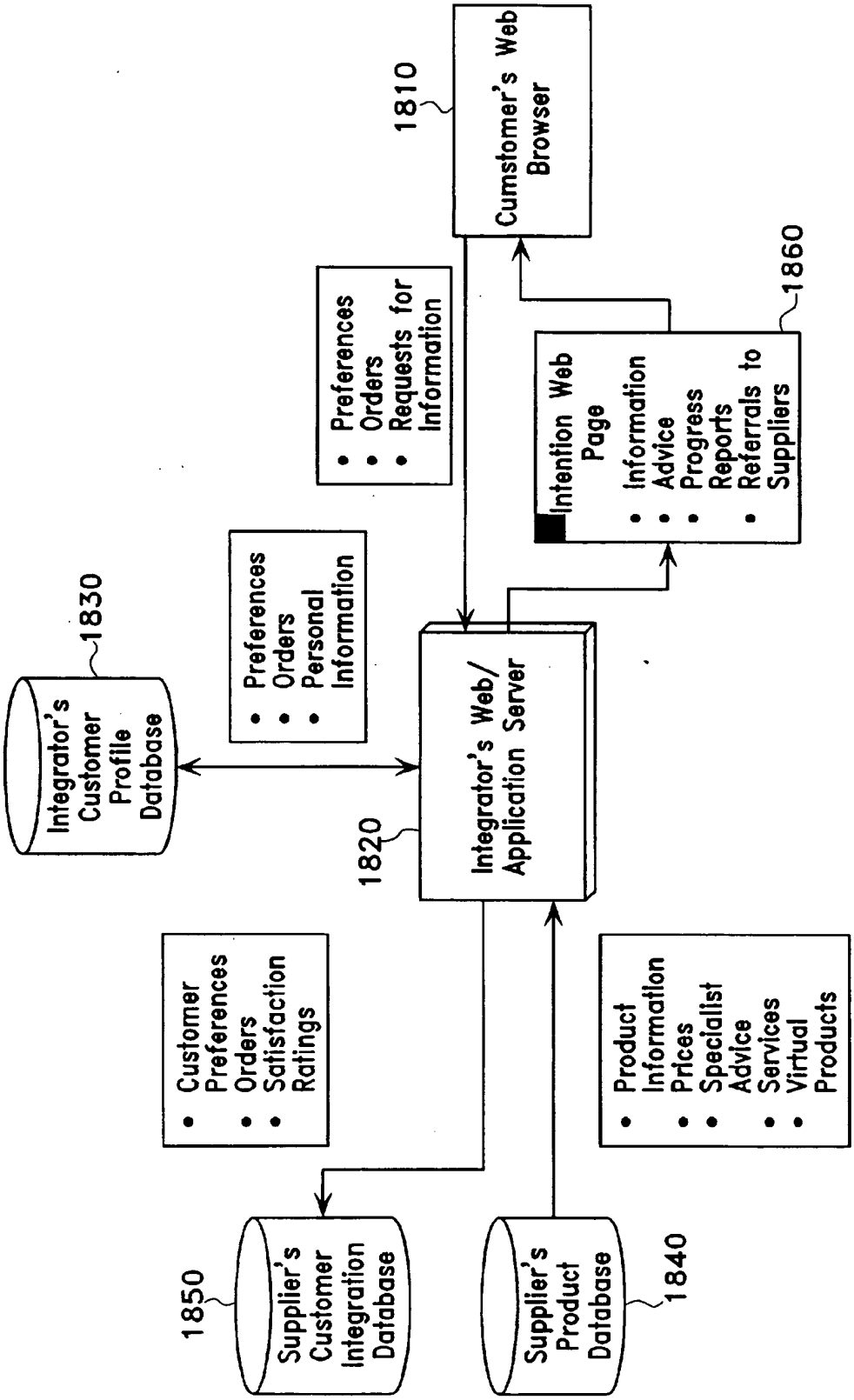

1810
Cumstomer's Web Browser
• Preferences
• Orders
• Requests for Information
1860
Intention Web Page
• Information
• Advice
• Progress Reports
• Referrals to Suppliers
1830
Integrator's Customer Profile Database
• Preferences
• Orders
• Personal Information
1820
Integrator's Web/ Application Server
• Customer Preferences
• Orders
• Satisfaction Ratings
• Product Information
• Prices
• Specialist Advice
• Services
• Virtual Products
1850
Supplier's Customer Integration Database
Supplier's Product Database
1840

Fig. 18

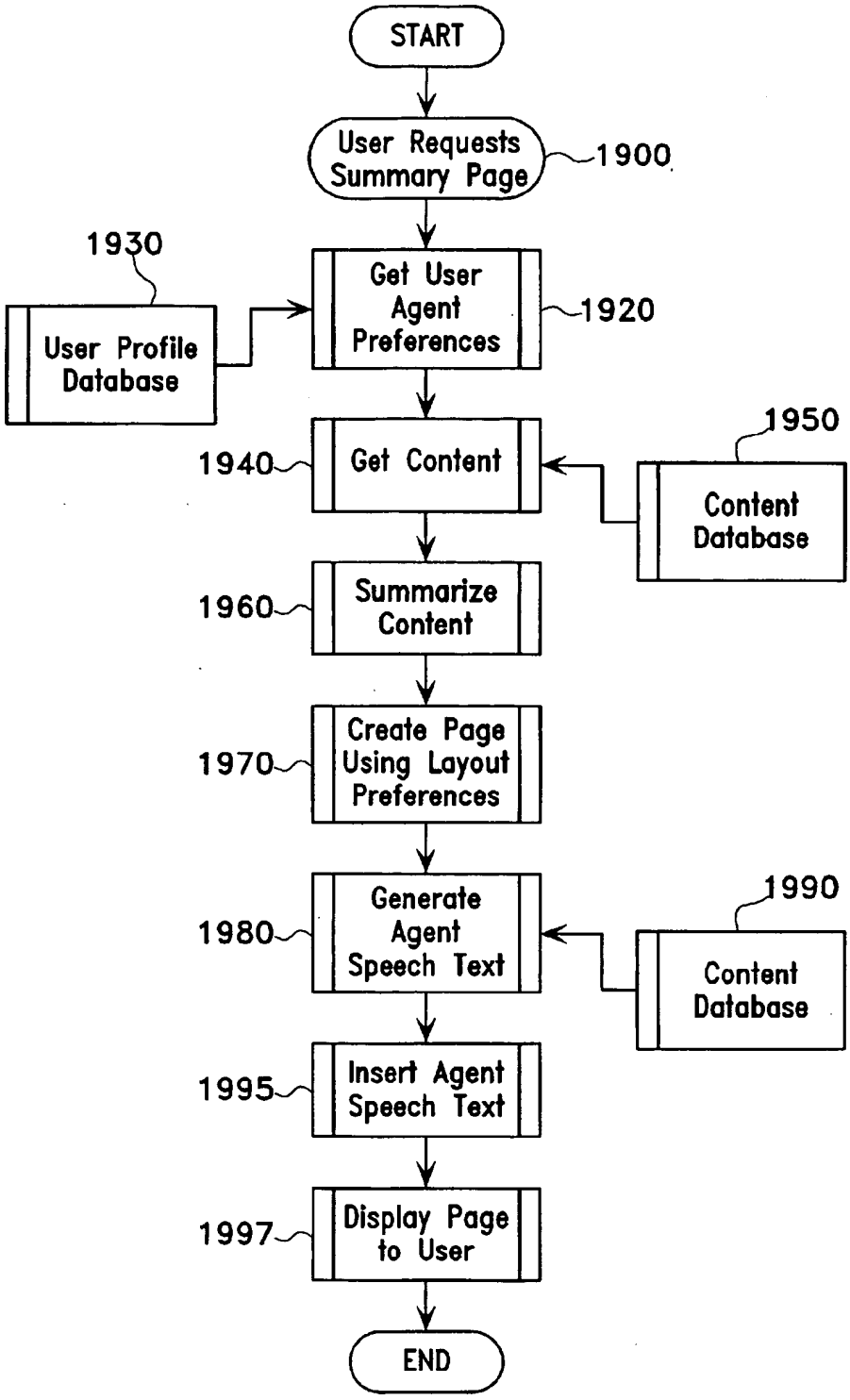

START
User Requests Summary Page
1900
1930
User Profile Database
Get User Agent Preferences
1920
1940
Get Content
1950
Content Database
1960
Summarize Content
1970
Create Page Using Layout Preferences
1980
Generate Agent Speech Text
1990
Content Database
1995
Insert Agent Speech Text
1997
Display Page to User
END

Fig. 19

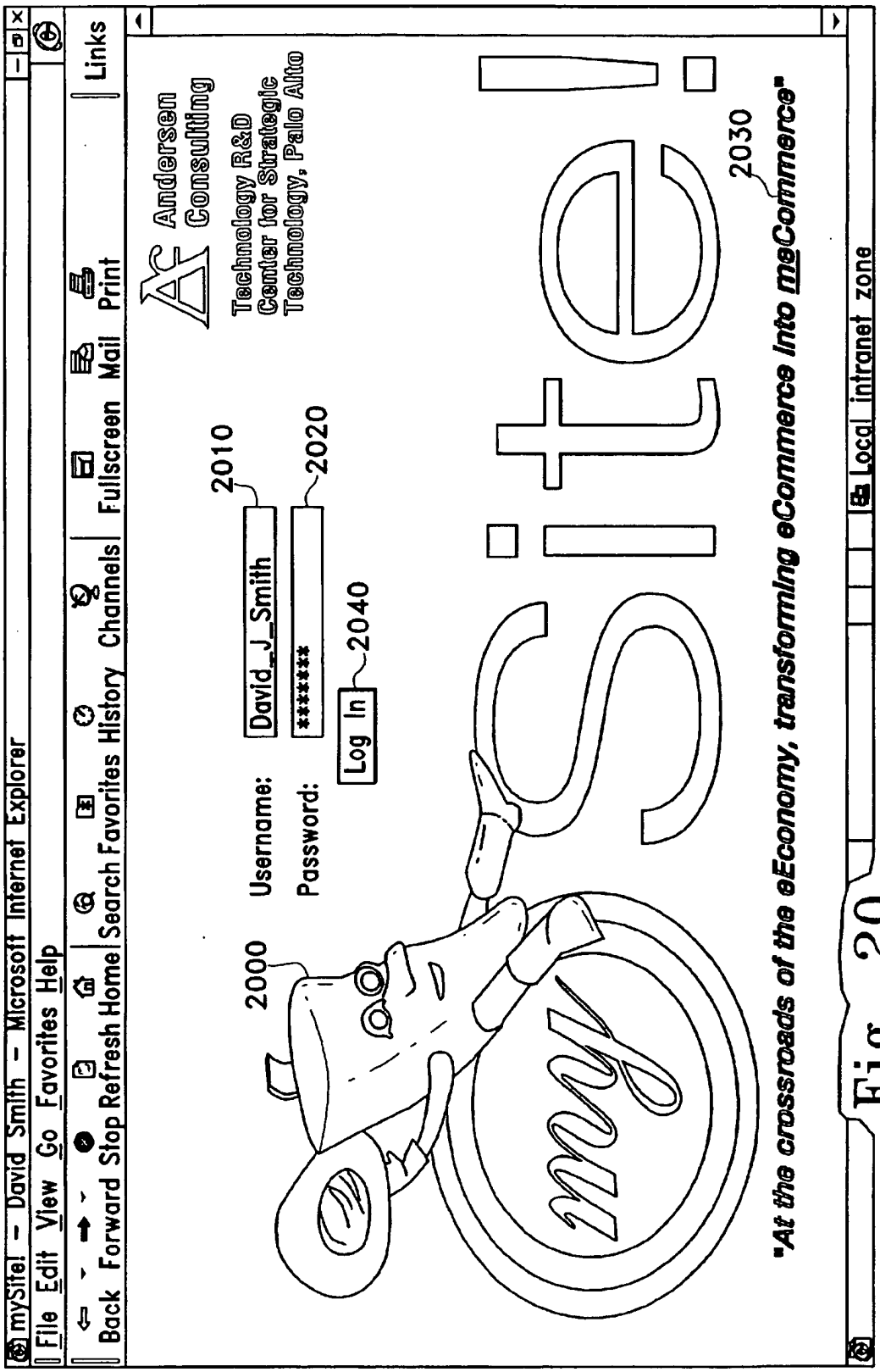

mySite! - David Smith - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Links
Andersen Consulting
Technology R&D
Center for Strategic
Technology, Palo Alto
2010
Username:
David_J_Smith
2020
Password:
*******
Log In
2040
2000
mySite!
2030
"At the crossroads of the eEconomy, transforming eCommerce into meCommerce"
Local intranet zone

Fig. 20

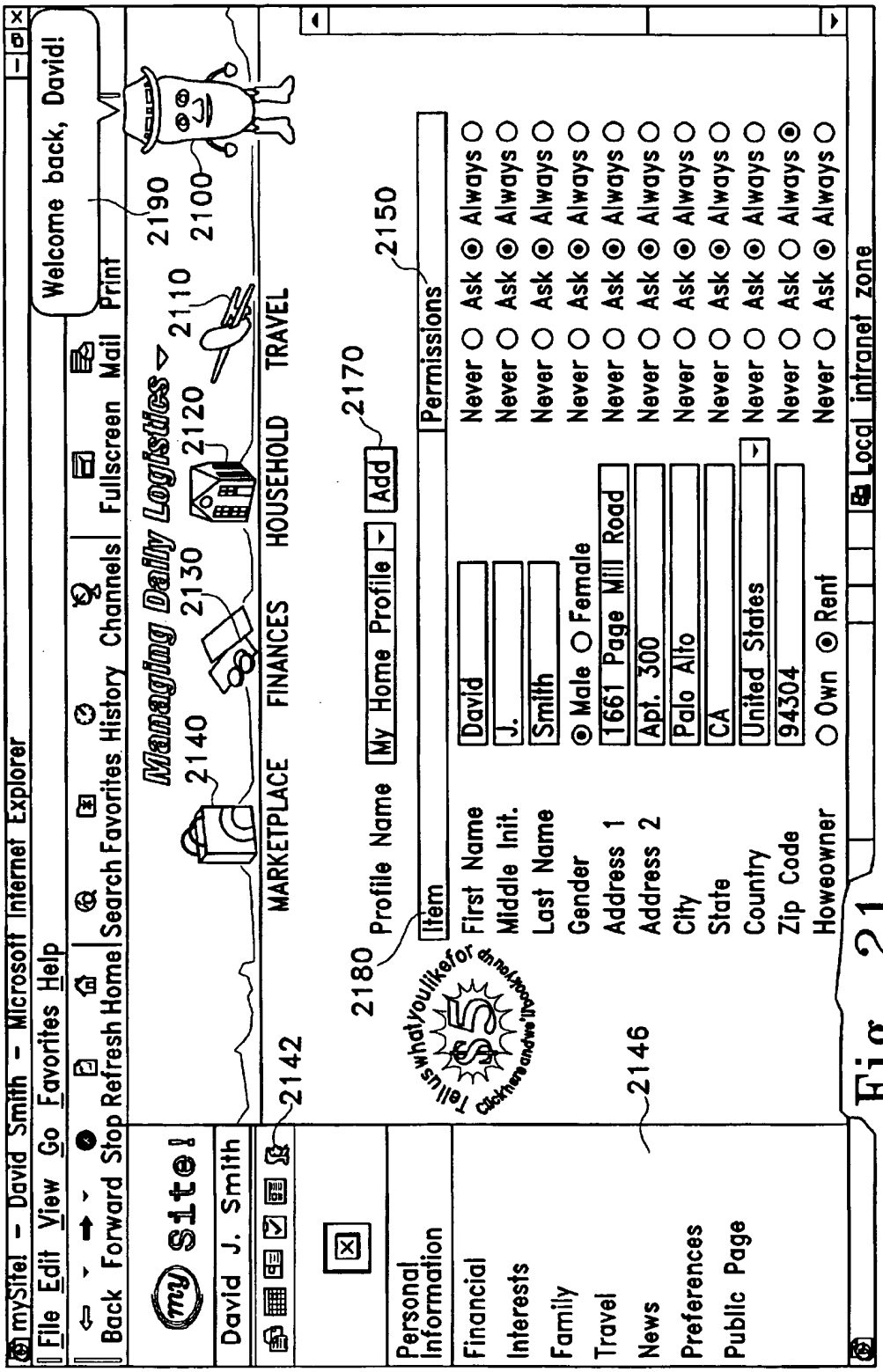

mySite! - David Smith - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print
Welcome back, David!
mySite!
David J. Smith
Managing Daily Logistics
MARKETPLACE FINANCES HOUSEHOLD TRAVEL
2140 2130 2120 2110 2100 2190
2142
2180 Profile Name My Home Profile Add 2170
2150
Item Permissions
Personal Information
Financial
Interests
Family
Travel
News
Preferences
Public Page
2146
Tell us what you like for dinner tonight and we'll cook it! Click here and we'll cook it!
First Name David Never Ask Always
Middle Init. J. Never Ask Always
Last Name Smith Never Ask Always
Gender Male Female Never Ask Always
Address 1 1661 Page Mill Road Never Ask Always
Address 2 Apt. 300 Never Ask Always
City Palo Alto Never Ask Always
State CA Never Ask Always
Country United States Never Ask Always
Zip Code 94304 Never Ask Always
Howeowner Own Rent Never Ask Always
Local intranet zone

Fig. 21

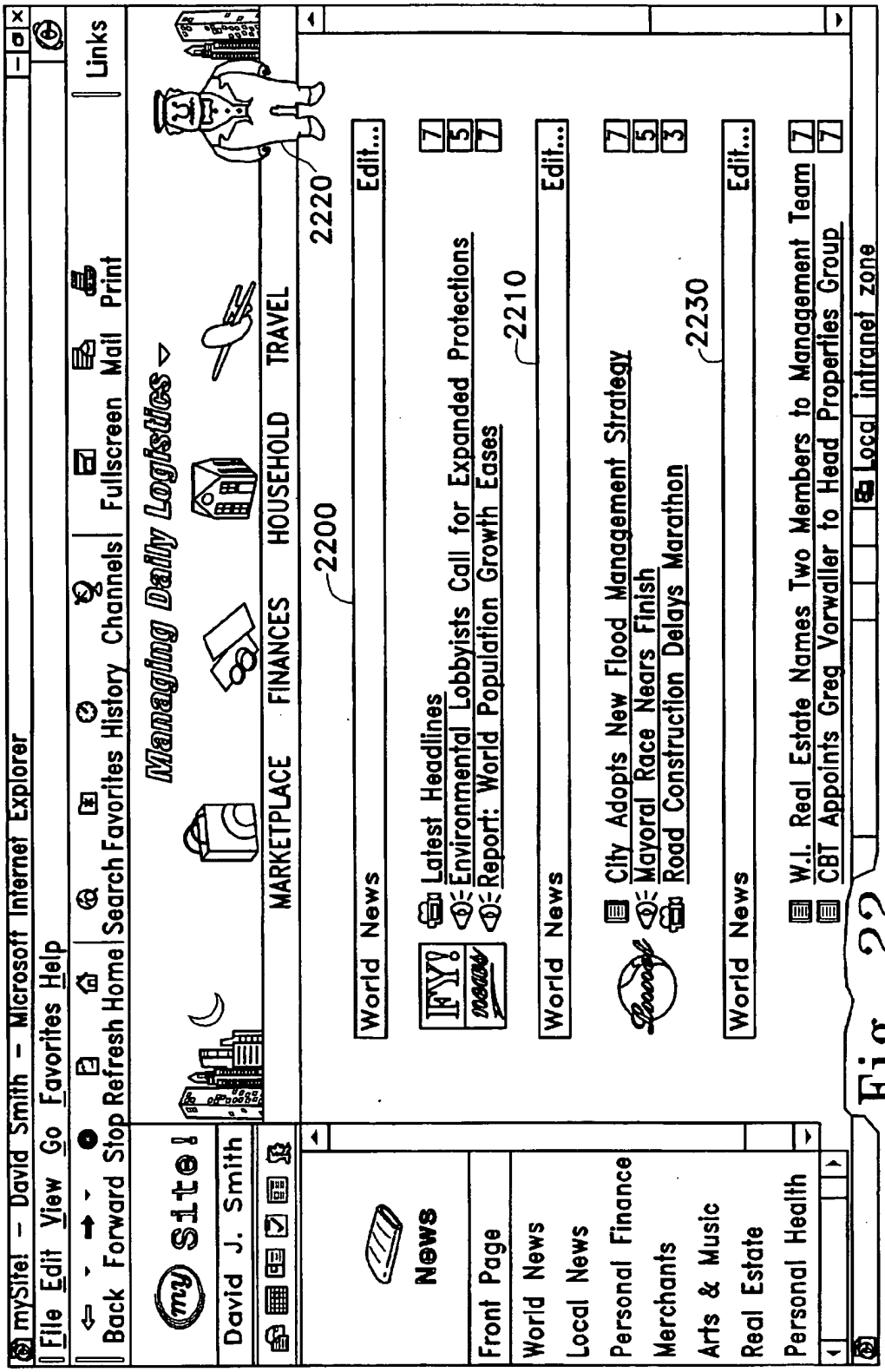

mySite! - David Smith - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Links
my Site!
David J. Smith
Managing Daily Logistics
MARKETPLACE FINANCES HOUSEHOLD TRAVEL
2200
2210
2220
2230
News
Front Page
World News
Local News
Personal Finance
Merchants
Arts & Music
Real Estate
Personal Health
World News
Edit...
Latest Headlines
Environmental Lobbyists Call for Expanded Protections
Report: World Population Growth Eases
City Adopts New Flood Management Strategy
Mayoral Race Nears Finish
Road Construction Delays Marathon
W.I. Real Estate Names Two Members to Management Team
CBT Appoints Greg Vorwaller to Head Properties Group
Local intranet zone

Fig. 22

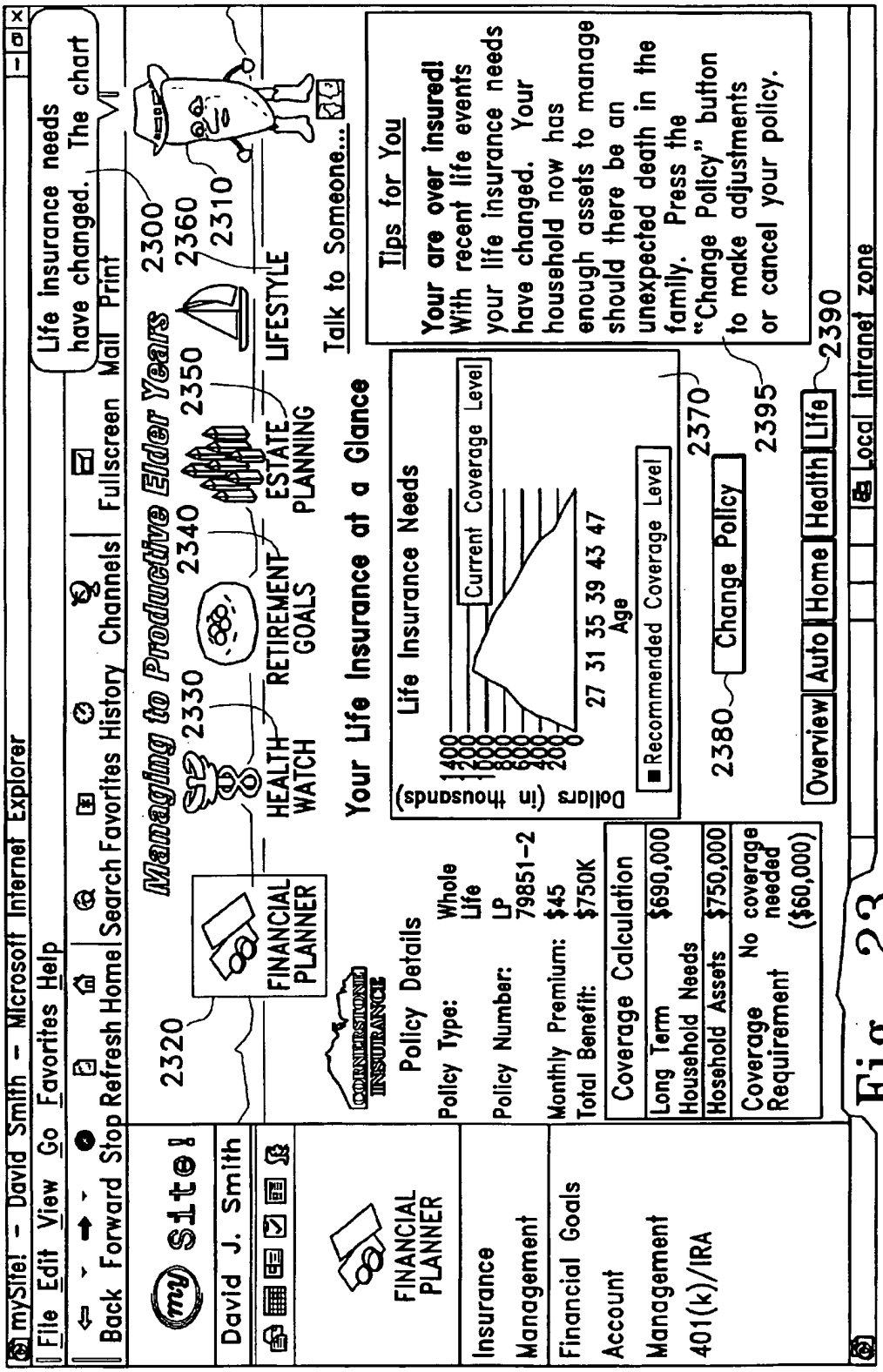

mySite! - David Smith - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print
mySite!
David J. Smith
FINANCIAL PLANNER
Insurance Management
Financial Goals Account Management
401(k)/IRA
2320
Managing to Productive Elder Years
2330
2340
2350
2360
2300
2310
HEALTH WATCH
RETIREMENT GOALS
ESTATE PLANNING
LIFESTYLE
Talk to Someone...
Your Life Insurance at a Glance
CONNECTICUT INSURANCE
Policy Details
Policy Type: Whole Life
Policy Number: LP 79851-2
Monthly Premium: $45
Total Benefit: $750K
Coverage Calculation
Long Term Household Needs $690,000
Hosehold Assets $750,000
Coverage Requirement No coverage needed ($60,000)
Life Insurance Needs
Dollars (in thousands)
1400
1200
1000
800
600
400
200
0
Current Coverage Level
27 31 35 39 43 47
Age
Recommended Coverage Level
2370
2380
Change Policy
2395
Tips for You
Your are over insured! With recent life events your life insurance needs have changed. Your household now has enough assets to manage should there be an unexpected death in the family. Press the "Change Policy" button to make adjustments or cancel your policy.
Life insurance needs have changed. The chart
Overview
Auto
Home
Health
Life
2390
Local intranet zone

Fig. 23

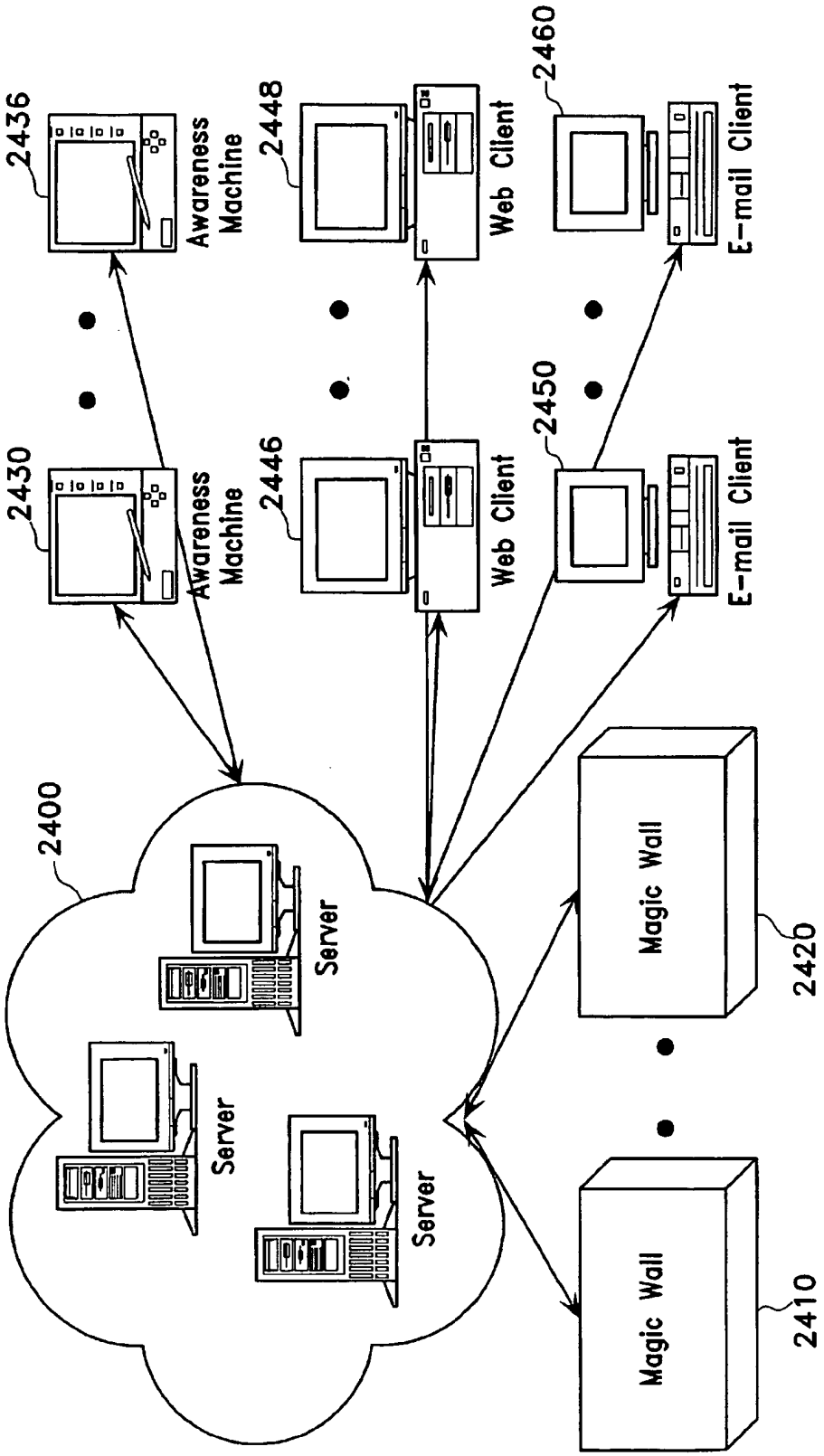

2436
Awareness Machine
2448
Web Client
2460
E-mail Client
2430
Awareness Machine
2446
Web Client
2450
E-mail Client
2400
Server
Server
Server
Magic Wall
2420
Magic Wall
2410

Fig. 24

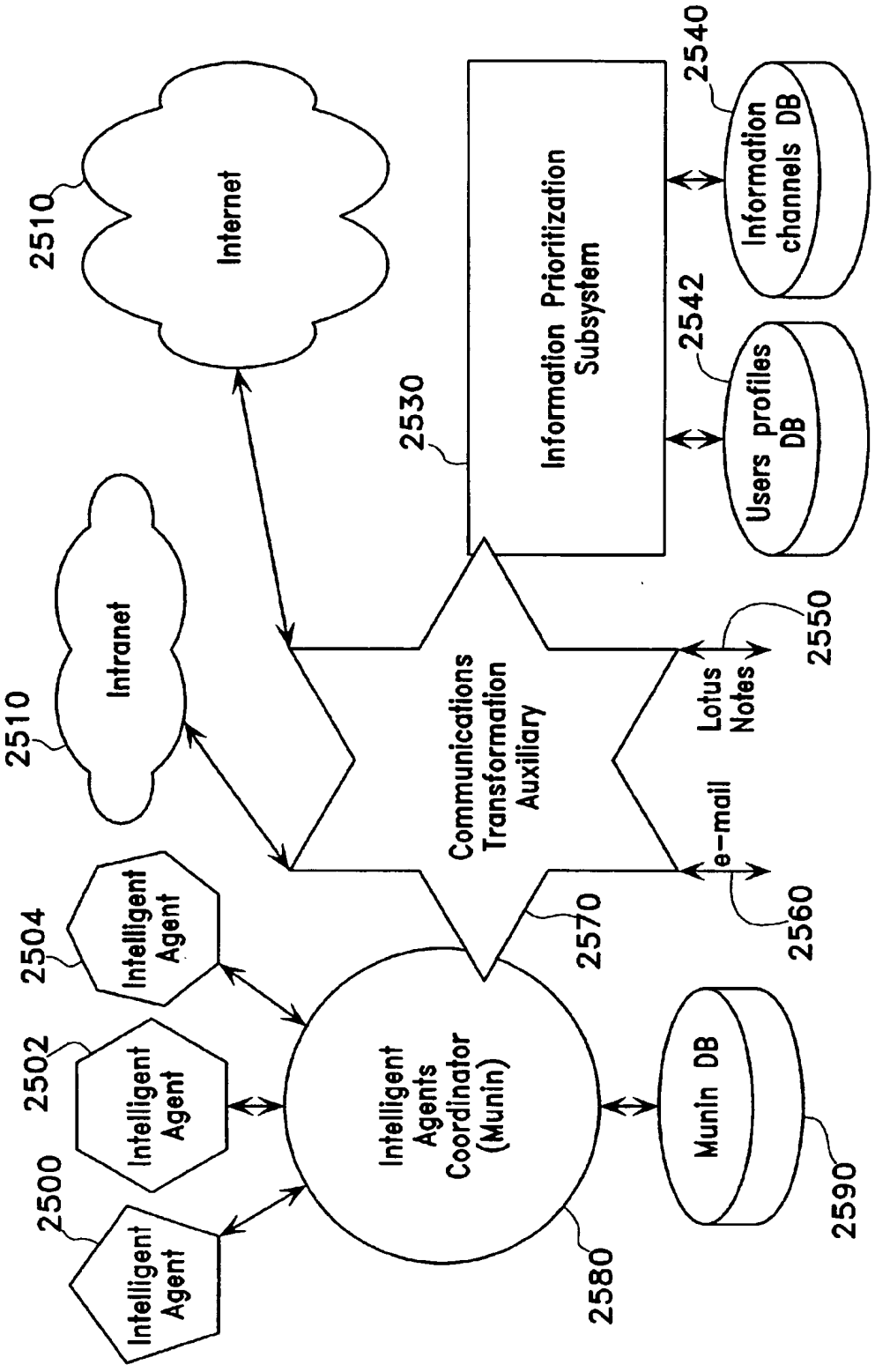

2510
Internet
2530
Information Prioritization Subsystem
2540
Information channels DB
2542
Users profiles DB
2510
Intranet
Communications Transformation Auxiliary
2550
Lotus Notes
e-mail
2560
2570
2504
Intelligent Agent
2502
Intelligent Agent
2500
Intelligent Agent
Intelligent Agents Coordinator (Munin)
2580
Munin DB
2590

Fig. 25

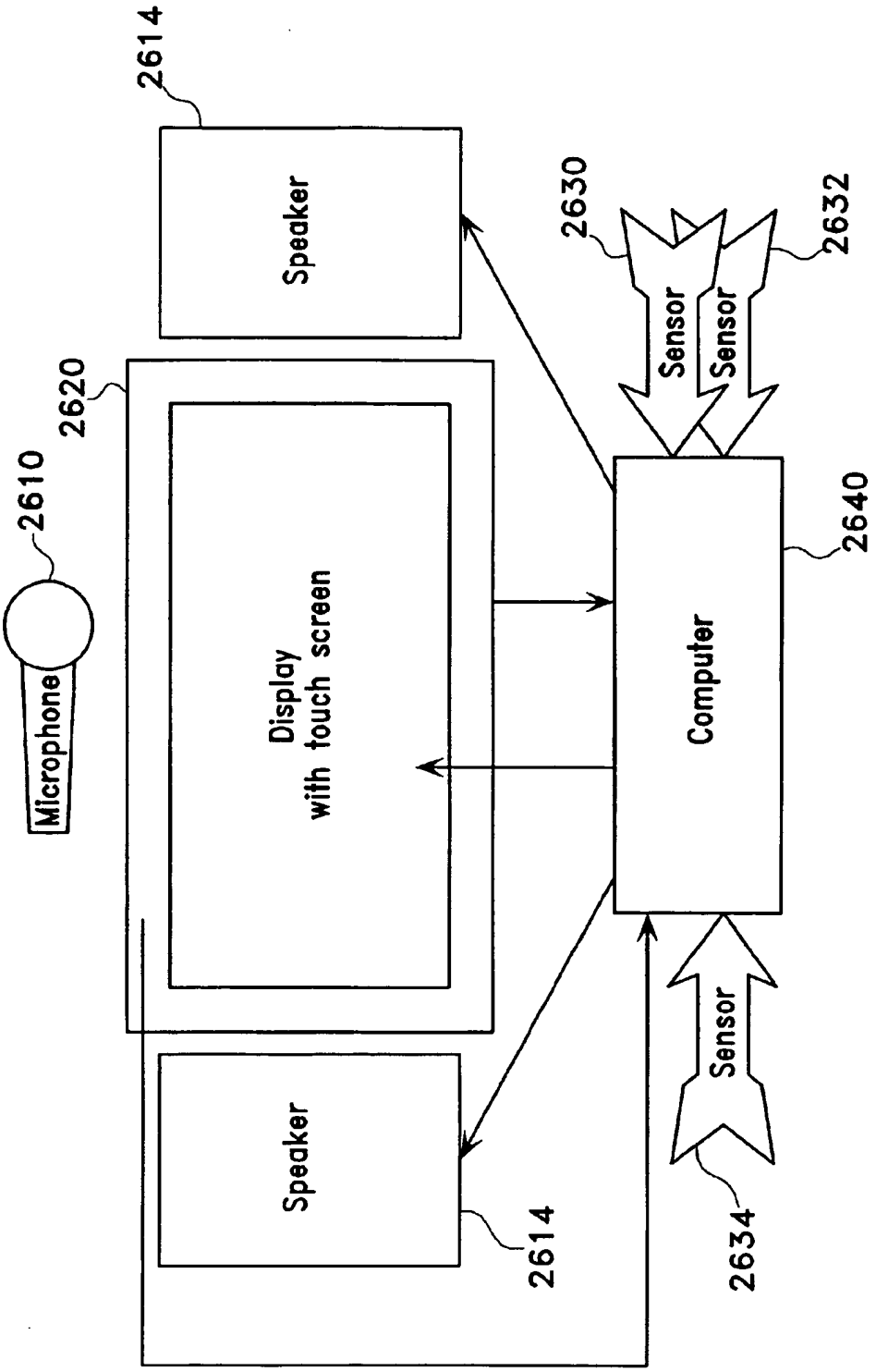

2614
Speaker
2630
2632
Sensor
Sensor
2620
2610
Microphone
Display
with touch screen
Computer
2640
Sensor
Speaker
2614
2634

Fig. 26

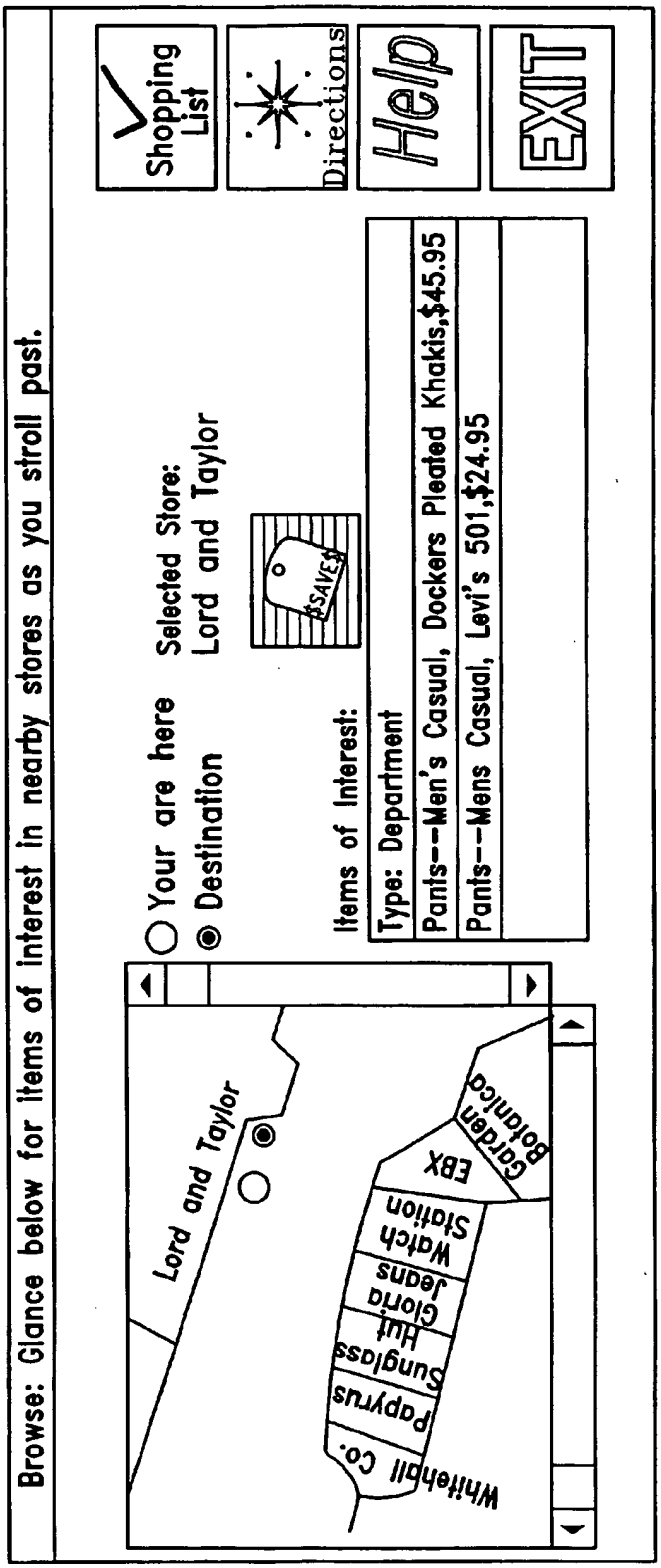

Browse: Glance below for items of interest in nearby stores as you stroll past.
Shopping List
Directions
Help
EXIT
Selected Store:
Lord and Taylor
Your are here
Destination
Items of Interest:
Type: Department
Pants--Men's Casual, Dockers Pleated Khakis,$45.95
Pants--Mens Casual, Levi's 501,$24.95
Lord and Taylor
Whitehall Co.
Papyrus
Sunglass Hut
Gloria Jeans
Watch Station
EBX
Garden Botanica

Fig. 27A

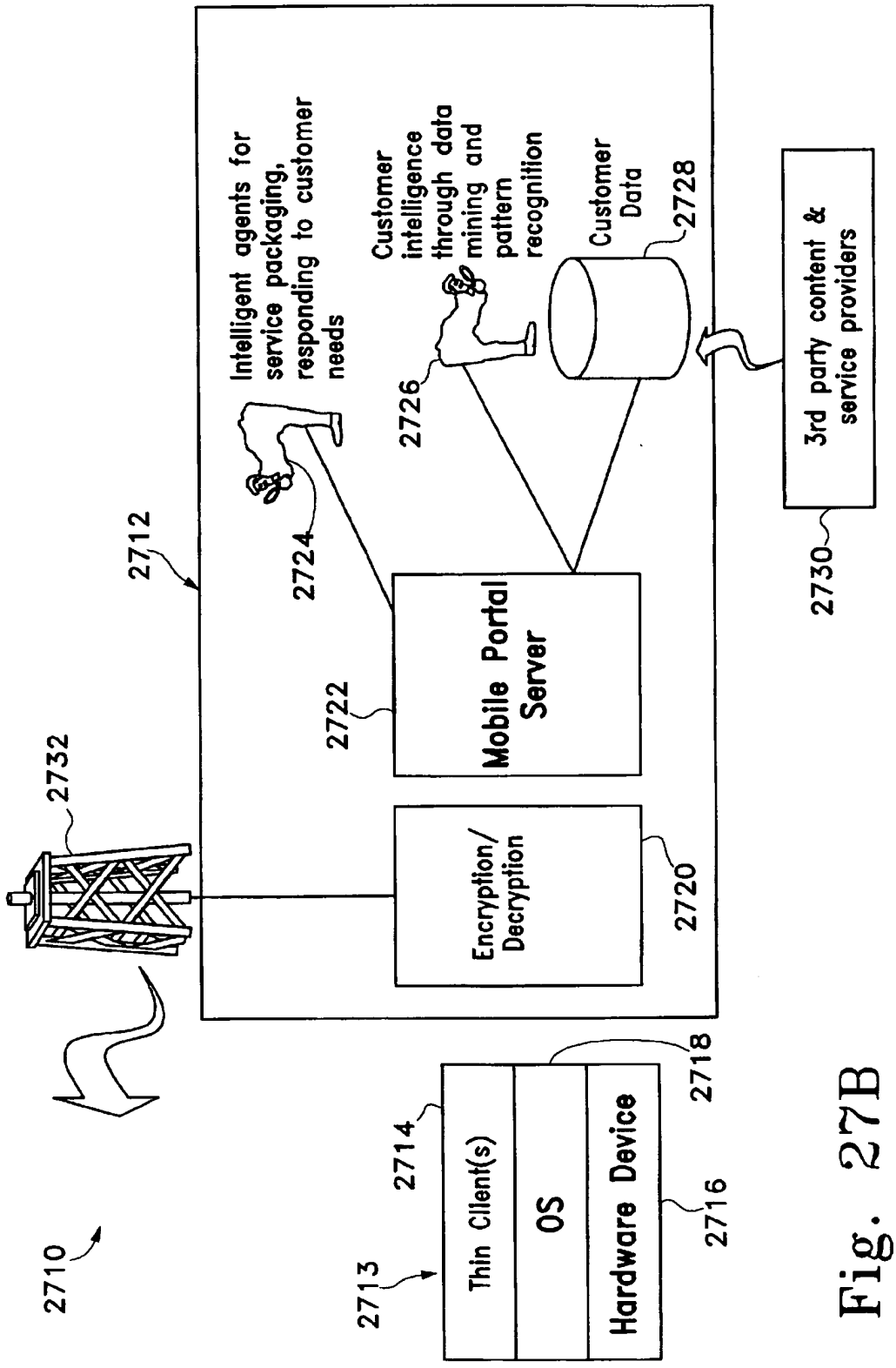

2710
2712
2713
2714
Thin Client(s)
OS
Hardware Device
2716
2718
2720
Encryption/ Decryption
2722
Mobile Portal Server
2724
Intelligent agents for service packaging, responding to customer needs
2726
Customer intelligence through data mining and pattern recognition
2728
Customer Data
2730
3rd party content & service providers
2732

Fig. 27B

# SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR MOBILE COMMUNICATION UTILIZING AN INTERFACE SUPPORT FRAMEWORK

## FIELD OF THE INVENTION

The present invention relates to agent based systems and more particularly to a mobile computing environment that accesses the Internet to obtain product information for a user through an interface support framework.

## BACKGROUND OF THE INVENTION

Computer assistance in all environments is increasingly necessary as computer technology becomes increasingly embedded in society. Mobile computing technology addresses this issue by allowing the individual to access computer related information at all times and in all environments.

One of the first major advances in mobile computer technology was the Personal Digital Assistant (PDA). A PDA allowed a user to access computer related information, yet fitted in the palm of the hand. Utilizing a PDA the user could organize personal affairs, write notes, calculate equations, and record contact numbers an address book. In addition, PDAs were usually capable of interfacing with a desktop computer, typically through a wire connection. The connection allowed the PDA to download information and upload information, with the desktop computer. Later developments gave the PDA wireless capabilities. The wireless capabilities allowed the PDA to interact with other computers that were not physically connected to the PDA.

Wireless PDAs could communicate with computers that were connected to the World Wide Web, and soon led to PDAs capable of Web browsing. One of the first companies to develop Web browsing capabilities for PDAs was Intercom.

Intercom's Falcon Mobile Server allowed PDAs with Web functions to directly connect to a host computer. Just by installing the software onto the host server, PDA terminals were able to access information through the World Wide Web.

Currently, more integration in mobile computing is desired. Nokia, an Irving Tex. company, has partially addressed the integration issue by developing the Nokia 9000 wireless voice phone. The Nokia 9000 includes a small keyboard, a specialized Web browser from microbrowser vendor Unwired Planet, Inc., and a small VGA monitor. Nokia worked with Ericsson Inc, Motorola Inc. and Unwired Planet to establish the Wireless Application Protocal (WAP), a standardized browser technology and server format. WAP gave manufacturers a standard way to put data capability into wireless phones, and allowed carriers to do more over-the-air management. For example, if a carrier wanted a field trial of a new data service, the carrier could implement the service on a server, deliver it to a phone through the microbrowser and adjust the service if they found the service unsatisfactory.

Prior Art FIG. 1A is a diagram of prior art mobile computing solutions based on web portal networks. In the Prior Art, the user **10** must deal separately with each participant of the network. In the Prior Art mobile computing solution, the user **10** utilizes an Internet service provider (ISP) **12** to gain access to a web portal **14**. The web portal **14** accesses third party services **16** which provide information directly to the user **10**. However, in addition to dealing



with the Internet Service Provider **12**, the user **10** must purchase the wireless device from the device manufactures or retailers **18**. In most cases the user **10** would also have to purchase the browser from the browser provider **20**. Generally, the user would have to pay the wireless communication cost, leading to the user needing to deal with the phone company **22**. And finally, any web purchases would lead to the user **10** needing to deal with the credit card company **24**. It is obvious that a coordinated and packaged service would be an ideal mobile computing solution. Furthermore, a coordinated and packaged service which made use of agents would be highly desired.

Agent based technology has become increasingly important for use with applications designed to interact with a user for performing various computer based tasks in foreground and background modes. Agent software comprises computer programs that are set on behalf of users to perform routine, tedious and time-consuming tasks. To be useful to an individual user, an agent must be personalized to the individual user's goals, habits and preferences. Thus, there exists a substantial requirement for the agent to efficiently and effectively acquire user-specific knowledge from the user and utilize it to perform tasks on behalf of the user.

The concept of agency, or the user of agents, is well established. An agent is a person authorized by another person, typically referred to as a principal, to act on behalf of the principal. In this manner the principal empowers the agent to perform any of the tasks that the principal is unwilling or unable to perform. For example, an insurance agent may handle all of the insurance requirements for a principal, or a talent agent may act on behalf of a performer to arrange concert dates.

With the advent of the computer, a new domain for employing agents has arrived. Significant advances in the realm of expert systems enable computer programs to act on behalf of computer users to perform routine, tedious and other time-consuming tasks. These computer programs are referred to as "software agents."

Moreover, there has been a recent proliferation of computer and communication networks. These networks permit a user to access vast amounts of information and services without, essentially, any geographical boundaries. Thus, a software agent has a rich environment to perform a large number of tasks on behalf of a user. For example, it is now possible for an agent to make an airline reservation, purchase the ticket, and have the ticket delivered directly to a user. Similarly, an agent could scan the Internet and obtain information ranging from the latest sports or news to a particular graduate thesis in applied physics. Current solutions fail to apply agent technology to provide targeted acquisition of information for a user's upcoming events.

## SUMMARY OF THE INVENTION

A system is disclosed that facilitates web-based information retrieval and display system. A wireless phone or similar hand-held wireless device with Internet Protocol capability is combined with other peripherals to provide a portable portal into the Internet. The wireless device prompts a user to input information of interest to the user. This information is transmitted a query to a service routine (running on a Web server). The service routine then queries the Web to find price, shipping and availability information from various Web suppliers. This information is formatted and displayed on the hand-held device's screen through an interface support framework. The user may then use the hand-held device to place an order interactively.

## DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, aspects and advantages are better understood from the following detailed description of a preferred embodiment of the invention with reference to the drawings, in which:

Prior Art FIG. **1A** is a diagram of Prior Art mobile computing solutions based on web portal networks;

FIG. **1** is a block diagram of a representative hardware environment in accordance with a preferred embodiment;

FIG. **2** is a flowchart of the system in accordance with a preferred embodiment;

FIG. **3** is a flowchart of a parsing unit of the system in accordance with a preferred embodiment;

FIG. **4** is a flowchart for pattern matching in accordance with a preferred embodiment;

FIG. **5** is a flowchart for a search unit in accordance with a preferred embodiment;

FIG. **6** is a flowchart for overall system processing in accordance with a preferred embodiment;

FIG. **7** is a flowchart of topic processing in accordance with a preferred embodiment;

FIG. **8** is a flowchart of meeting record processing in accordance with a preferred embodiment;

FIG. **9** is a block diagram of process flow of a pocket bargain finder in accordance with a preferred embodiment;

FIGS. **10A** and **10B** are a block diagram and flowchart depicting the logic associated with creating a customized content web page in accordance with a preferred embodiment;

FIG. **11** is a flowchart depicting the detailed logic associated with retrieving user-centric content in accordance with a preferred embodiment;

FIG. **12** is a data model of a user profile in accordance with a preferred embodiment;

FIG. **13** is a persona data model in accordance with a preferred embodiment;

FIG. **14** is an intention data model in accordance with a preferred embodiment;

FIG. **15** is a flowchart of the processing for generating an agent's current statistics in accordance with a preferred embodiment;

FIG. **16** is a flowchart of the logic that determines the personalized product rating for a user in accordance with a preferred embodiment;

FIG. **17** is a flowchart of the logic for accessing the centrally stored profile in accordance with a preferred embodiment;

FIG. **18** is a flowchart of the interaction logic between a user and the integrator for a particular supplier in accordance with a preferred embodiment;

FIG. **19** is a flowchart of the agent processing for generating a verbal summary in accordance with a preferred embodiment;

FIG. **20** illustrates a display login in accordance with a preferred embodiment;

FIG. **21** illustrates a managing daily logistics display in accordance with a preferred embodiment;

FIG. **22** illustrates a user main display in accordance with a preferred embodiment;

FIG. **23** illustrates an agent interaction display in accordance with a preferred embodiment;

FIG. **24** is a block diagram of an active knowledge management system in accordance with a preferred embodiment;

FIG. **25** is a block diagram of a back end server in accordance with a preferred embodiment;

FIG. **26** is a flow chart illustrating how the hardware and software of one embodiment of the present invention operates;

FIG. **27A** illustrates a display of the browser mode in accordance with a preferred embodiment; and

FIG. **27B** is an illustration of a Mobile Portal platform in accordance with a preferred embodiment.

## DETAILED DESCRIPTION

A preferred embodiment of a system in accordance with the present invention is preferably practiced in the context of a personal computer such as an IBM compatible personal computer, Apple Macintosh computer or UNIX based workstation. A representative hardware environment is depicted in FIG. **1**, which illustrates a typical hardware configuration of a workstation in accordance with a preferred embodiment having a central processing unit **110**, such as a microprocessor, and a number of other units interconnected via a system bus **112**. The workstation shown in FIG. **1** includes a Random Access Memory (RAM) **114**, Read Only Memory (ROM) **116**, an I/O adapter **118** for connecting peripheral devices such as disk storage units **120** to the bus **112**, a user interface adapter **122** for connecting a keyboard **124**, a mouse **126**, a speaker **128**, a microphone **132**, and/or other user interface devices such as a touch screen (not shown) to the bus **112**, communication adapter **134** for connecting the workstation to a communication network (e.g., a data processing network) and a display adapter **136** for connecting the bus **112** to a display device **138**. The workstation typically has resident thereon an operating system such as the Microsoft Windows NT or Windows/95 Operating System (OS), the IBM OS/2 operating system, the MAC OS, or UNIX operating system. Those skilled in the art will appreciate that the present invention may also be implemented on platforms and operating systems other than those mentioned.

A preferred embodiment is written using JAVA, C, and the C++ language and utilizes object oriented programming methodology. Object oriented programming (OOP) has become increasingly used to develop complex applications. As OOP moves toward the mainstream of software design and development, various software solutions require adaptation to make use of the benefits of OOP. A need exists for these principles of OOP to be applied to a messaging interface of an electronic messaging system such that a set of OOP classes and objects for the messaging interface can be provided. OOP is a process of developing computer software using objects, including the steps of analyzing the problem, designing the system, and constructing the program. An object is a software package that contains both data and a collection of related structures and procedures. Since it contains both data and a collection of structures and procedures, it can be visualized as a self-sufficient component that does not require other additional structures, procedures or data to perform its specific task. OOP, therefore, views a computer program as a collection of largely autonomous components, called objects, each of which is responsible for a specific task. This concept of packaging data, structures, and procedures together in one component or module is called encapsulation.

In general, OOP components are reusable software modules which present an interface that conforms to an object model and which are accessed at run-time through a component integration architecture. A component integration

architecture is a set of architecture mechanisms which allow software modules in different process spaces to utilize each others capabilities or functions. This is generally done by assuming a common component object model on which to build the architecture.

It is worthwhile to differentiate between an object and a class of objects at this point. An object is a single instance of the class of objects, which is often just called a class. A class of objects can be viewed as a blueprint, from which many objects can be formed.

OOP allows the programmer to create an object that is a part of another object. For example, the object representing a piston engine is said to have a composition-relationship with the object representing a piston. In reality, a piston engine comprises a piston, valves and many other components; the fact that a piston is an element of a piston engine can be logically and semantically represented in OOP by two objects.

OOP also allows creation of an object that "depends from" another object. If there are two objects, one representing a piston engine and the other representing a piston engine wherein the piston is made of ceramic, then the relationship between the two objects is not that of composition. A ceramic piston engine does not make up a piston engine. Rather it is merely one kind of piston engine that has one more limitation than the piston engine; its piston is made of ceramic. In this case, the object representing the ceramic piston engine is called a derived object, and it inherits all of the aspects of the object representing the piston engine and adds further limitation or detail to it. The object representing the ceramic piston engine "depends from" the object representing the piston engine. The relationship between these objects is called inheritance.

When the object or class representing the ceramic piston engine inherits all of the aspects of the objects representing the piston engine, it inherits the thermal characteristics of a standard piston defined in the piston engine class. However, the ceramic piston engine object overrides these ceramic specific thermal characteristics, which are typically different from those associated with a metal piston. It skips over the original and uses new functions related to ceramic pistons. Different kinds of piston engines have different characteristics, but may have the same underlying functions associated with it (e.g., how many pistons in the engine, ignition sequences, lubrication, etc.). To access each of these functions in any piston engine object, a programmer would call the same functions with the same names, but each type of piston engine may have different/overriding implementations of functions behind the same name. This ability to hide different implementations of a function behind the same name is called polymorphism and it greatly simplifies communication among objects.

With the concepts of composition-relationship, encapsulation, inheritance and polymorphism, an object can represent just about anything in the real world. In fact, our logical perception of the reality is the only limit on determining the kinds of things that can become objects in object-oriented software. Some typical categories are as follows:

- Objects can represent physical objects, such as automobiles in a traffic-flow simulation, electrical components in a circuit-design program, countries in an economics model, or aircraft in an air-traffic-control system.
- Objects can represent elements of the computer-user environment such as windows, menus or graphics objects.
- An object can represent an inventory, such as a personnel file or a table of the latitudes and longitudes of cities.
- An object can represent user-defined data types such as time, angles, and complex numbers, or points on the plane.

With this enormous capability of an object to represent just about any logically separable matters, OOP allows the software developer to design and implement a computer program that is a model of some aspects of reality, whether that reality is a physical entity, a process, a system, or a composition of matter. Since the object can represent anything, the software developer can create an object which can be used as a component in a larger software project in the future.

If 90% of a new OOP software program consists of proven, existing components made from preexisting reusable objects, then only the remaining 10% of the new software project has to be written and tested from scratch. Since 90% already came from an inventory of extensively tested reusable objects, the potential domain from which an error could originate is 10% of the program. As a result, OOP enables software developers to build objects out of other, previously built, objects.

This process closely resembles complex machinery being built out of assemblies and sub-assemblies. OOP technology, therefore, makes software engineering more like hardware engineering in that software is built from existing components, which are available to the developer as objects. All this adds up to an improved quality of the software as well as an increased speed of its development.

Programming languages are beginning to fully support the OOP principles, such as encapsulation, inheritance, polymorphism, and composition-relationship. With the advent of the C++ language, many commercial software developers have embraced OOP. C++ is an OOP language that offers a fast, machine-executable code. Furthermore, C++ is suitable for both commercial-application and systems-programming projects. For now, C++ appears to be the most popular choice among many OOP programmers, but there is a host of other OOP languages, such as Smalltalk, common lisp object system (CLOS), and Eiffel. Additionally, OOP capabilities are being added to more traditional popular computer programming languages such as Pascal.

The benefits of object classes can be summarized, as follows:

- Objects and their corresponding classes break down complex programming problems into many smaller, simpler problems.
- Encapsulation enforces data abstraction through the organization of data into small, independent objects that can communicate with each other. Encapsulation protects the data in an object from accidental damage, but allows other objects to interact with that data by calling the object's member functions and structures.
- Subclassing and inheritance make it possible to extend and modify objects through deriving new kinds of objects from the standard classes available in the system. Thus, new capabilities are created without having to start from scratch.
- Polymorphism and multiple inheritance make it possible for different programmers to mix and match characteristics of many different classes and create specialized objects that can still work with related objects in predictable ways.
- Class hierarchies and containment hierarchies provide a flexible mechanism for modeling real-world objects and the relationships among them.

Libraries of reusable classes are useful in many situations, but they also have some limitations. For example:

Complexity. In a complex system, the class hierarchies for related classes can become extremely confusing, with many dozens or even hundreds of classes.

Flow of control. A program written with the aid of class libraries is still responsible for the flow of control (i.e., it must control the interactions among all the objects created from a particular library). The programmer has to decide which functions to call at what times for which kinds of objects.

Duplication of effort. Although class libraries allow programmers to use and reuse many small pieces of code, each programmer puts those pieces together in a different way. Two different programmers can use the same set of class libraries to write two programs that do exactly the same thing but whose internal structure (i.e., design) may be quite different, depending on hundreds of small decisions each programmer makes along the way. Inevitably, similar pieces of code end up doing similar things in slightly different ways and do not work as well together as they should.

Class libraries are very flexible. As programs grow more complex, more programmers are forced to reinvent basic solutions to basic problems over and over again. A relatively new extension of the class library concept is to have a framework of class libraries. This framework is more complex and consists of significant collections of collaborating classes that capture both the small scale patterns and major mechanisms that implement the common requirements and design in a specific application domain. They were first developed to free application programmers from the chores involved in displaying menus, windows, dialog boxes, and other standard user interface elements for personal computers.

Frameworks also represent a change in the way programmers think about the interaction between the code they write and code written by others. In the early days of procedural programming, the programmer called libraries provided by the operating system to perform certain tasks, but basically the program executed down the page from start to finish, and the programmer was solely responsible for the flow of control. This was appropriate for printing out paychecks, calculating a mathematical table, or solving other problems with a program that executed in just one way.

The development of graphical user interfaces began to turn this procedural programming arrangement inside out. These interfaces allow the user, rather than program logic, to drive the program and decide when certain actions should be performed. Today, most personal computer software accomplishes this by means of an event loop which monitors the mouse, keyboard, and other sources of external events and calls the appropriate parts of the programmer's code according to actions that the user performs. The programmer no longer determines the order in which events occur. Instead, a program is divided into separate pieces that are called at unpredictable times and in an unpredictable order. By relinquishing control in this way to users, the developer creates a program that is much easier to use. Nevertheless, individual pieces of the program written by the developer still call libraries provided by the operating system to accomplish certain tasks, and the programmer must still determine the flow of control within each piece after being called by the event loop. Application code still "sits on top of" the system.

Even event loop programs require programmers to write a lot of code that should not need to be written separately for every application. The concept of an application framework carries the event loop concept further. Instead of dealing with all the nuts and bolts of constructing basic menus, windows, and dialog boxes and then making these things all work together, programmers using application frameworks start with working application code and basic user interface elements in place. Subsequently, they build from there by replacing some of the generic capabilities of the framework with the specific capabilities of the intended application.

Application frameworks reduce the total amount of code that a programmer has to write from scratch. However, because the framework is really a generic application that displays windows, supports copy and paste, and so on, the programmer can also relinquish control to a greater degree than event loop programs permit. The framework code takes care of almost all event handling and flow of control, and the programmer's code is called only when the framework needs it (e.g., to create or manipulate a proprietary data structure).

A programmer writing a framework program not only relinquishes control to the user (as is also true for event loop programs), but also relinquishes the detailed flow of control within the program to the framework. This approach allows the creation of more complex systems that work together in interesting ways, as opposed to isolated programs, having custom code, being created over and over again for similar problems.

Thus, as is explained above, a framework basically is a collection of cooperating classes that make up a reusable design solution for a given problem domain. It typically includes objects that provide default behavior (e.g., for menus and windows), and programmers use it by inheriting some of that default behavior and overriding other behavior so that the framework calls application code at the appropriate times.

There are three main differences between frameworks and class libraries:

Behavior versus protocol. Class libraries are essentially collections of behaviors that you can call when you want those individual behaviors in your program. A framework, on the other hand, provides not only behavior but also the protocol or set of rules that govern the ways in which behaviors can be combined, including rules for what a programmer is supposed to provide versus what the framework provides.

Call versus override. With a class library, the code the programmer instantiates objects and calls their member functions. It's possible to instantiate and call objects in the same way with a framework (i.e., to treat the framework as a class library), but to take full advantage of a framework's reusable design, a programmer typically writes code that overrides and is called by the framework. The framework manages the flow of control among its objects. Writing a program involves dividing responsibilities among the various pieces of software that are called by the framework rather than specifying how the different pieces should work together.

Implementation versus design. With class libraries, programmers reuse only implementations, whereas with frameworks, they reuse design. A framework embodies the way a family of related programs or pieces of software work. It represents a generic design solution that can be adapted to a variety of specific problems in a given domain. For example, a single framework can embody the way a user interface works, even though two different user interfaces created with the same framework might solve quite different interface problems.

Thus, through the development of frameworks for solutions to various problems and programming tasks, significant reductions in the design and development effort for software can be achieved. A preferred embodiment of the invention utilizes HyperText Markup Language (HTML) to implement documents on the Internet together with a general-purpose secure communication protocol for a transport medium between the client and the Newco. HTTP or other protocols could be readily substituted for HTML without undue experimentation. Information on these products is available in T. Berners-Lee, D. Connoly, "RFC 1866: Hypertext Markup Language—2.0" (November 1995); and R. Fielding, H, Frystyk, T. Berners-Lee, J. Gettys and J. C. Mogul, "Hypertext Transfer Protocol—HTTP/1.1: HTTP Working Group Internet Draft" (May 2, 1996). HTML is a simple data format used to create hypertext documents that are portable from one platform to another. HTML documents are SGML documents with generic semantics that are appropriate for representing information from a wide range of domains. HTML has been in use by the World-Wide Web global information initiative since 1990. HTML is an application of ISO Standard 8879:1986 Information Processing Text and Office Systems; Standard Generalized Markup Language (SGML).

To date, Web development tools have been limited in their ability to create dynamic Web applications which span from client to server and interoperate with existing computing resources. Until recently, HTML has been the dominant technology used in development of Web-based solutions. However, HTML has proven to be inadequate in the following areas:

- Poor performance;
- Restricted user interface capabilities;
- Can only produce static Web pages;
- Lack of interoperability with existing applications and data; and
- Inability to scale.

Sun Microsystem's Java language solves many of the client-side problems by:

- Improving performance on the client side;
- Enabling the creation of dynamic, real-time Web applications; and
- Providing the ability to create a wide variety of user interface components.

With Java, developers can create robust User Interface (UI) components. Custom "widgets" (e.g., real-time stock tickers, animated icons, etc.) can be created, and client-side performance is improved. Unlike HTML, Java supports the notion of client-side validation, offloading appropriate processing onto the client for improved performance. Dynamic, real-time Web pages can be created. Using the above-mentioned custom UI components, dynamic Web pages can also be created.

Sun's Java language has emerged as an industry-recognized language for "programming the Internet." Sun defines Java as: "a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high-performance, multithreaded, dynamic, buzzword-compliant, general-purpose programming language. Java supports programming for the Internet in the form of platform-independent Java applets." Java applets are small, specialized applications that comply with Sun's Java Application Programming Interface (API) allowing developers to add "interactive content" to Web documents (e.g., simple animations, page adornments, basic games, etc.). Applets execute within a Java-compatible browser (e.g., Netscape Navigator) by copying code from the server to client. From a language standpoint, Java's core feature set is based on C++. Sun's Java literature states that Java is basically "C++, with extensions from Objective C for more dynamic method resolution".

Another technology that provides similar function to JAVA is provided by Microsoft and ActiveX Technologies, to give developers and Web designers wherewithal to build dynamic content for the Internet and personal computers. ActiveX includes tools for developing animation, 3-D virtual reality, video and other multimedia content. The tools use Internet standards, work on multiple platforms, and are being supported by over 100 companies. The group's building blocks are called ActiveX Controls, small, fast components that enable developers to embed parts of software in hypertext markup language (HTML) pages. ActiveX Controls work with a variety of programming languages including Microsoft Visual C++, Borland Delphi, Microsoft Visual Basic programming system and, in the future, Microsoft's development tool for Java, code named "Jakarta." ActiveX Technologies also includes ActiveX Server Framework, allowing developers to create server applications. One of ordinary skill in the art readily recognizes that ActiveX could be substituted for JAVA without undue experimentation to practice the invention.

In accordance with a preferred embodiment, BackgroundFinder (BF) is implemented as an agent responsible for preparing an individual for an upcoming meeting by helping him/her retrieve relevant information about the meeting from various sources. BF receives input text in character form indicative of the target meeting. The input text is generated in accordance with a preferred embodiment by a calendar program that includes the time of the meeting. As the time of the meeting approaches, the calendar program is queried to obtain the text of the target event and that information is utilized as input to the agent. Then, the agent parses the input meeting text to extract its various components such as title, body, participants, location, time etc. The system also performs pattern matching to identify particular meeting fields in a meeting text. This information is utilized to query various sources of information on the web and obtain relevant stories about the current meeting to send back to the calendaring system. For example, if an individual has a meeting with Netscape and Microsoft to talk about their disputes, and would obtain this initial information from the calendaring system. It will then parse out the text to realize that the companies in the meeting are "Netscape" and "Microsoft" and the topic is "disputes." Then, the system queries the web for relevant information concerning the topic. Thus, in accordance with an objective of the invention, the system updates the calendaring system and eventually the user with the best information it can gather to prepare the user for the target meeting. In accordance with a preferred embodiment, the information is stored in a file that is obtained via selection from a link imbedded in the calendar system.

### Program Organization

A computer program in accordance with a preferred embodiment is organized in five distinct modules: BF.Main, BF.Parse, Background Finder.Error, BF.PatternMatching and BF.Search. There is also a frmMain which provides a user interface used only for debugging purposes. The executable programs in accordance with a preferred embodiment never execute with the user interface and should only return to the calendaring system through Microsoft's Winsock control. A preferred embodiment of the system

executes in two different modes which can be specified under the command line sent to it by the calendaring system. When the system runs in simple mode, it executes a keyword query to submit to external search engines. When executed in complex mode, the system performs pattern matching before it forms a query to be sent to a search engine.

## Data Structures

The system in accordance with a preferred embodiment utilizes three user defined structures:

1. TMeetingRecord;
2. TPatternElement; and
3. TPatternRecord.

The user-defined structure, tMeetingRecord, is used to store all the pertinent information concerning a single meeting. This info includes userID, an original description of the meeting, the extracted list of keywords from the title and body of meeting etc. It is important to note that only one meeting record is created per instance of the system in accordance with a preferred embodiment. This is because each time the system is spawned to service an upcoming meeting, it is assigned a task to retrieve information for only one meeting. Therefore, the meeting record created corresponds to the current meeting examined. ParseMeetingText populates this meeting record and it is then passed around to provide information about the meeting to other functions. If GoPatternMatch can bind any values to a particular meeting field, the corresponding entries in the meeting record is also updated. The structure of tMeetingRecord with each field described in parentheses is provided below in accordance with a preferred embodiment.

| A.1.1.1.1.1 | Public Type tMeetingRecord |
|---|---|
| sUserID As String | (user id given by Munin) |
| sTitleOrig As String | (original non stop listed title we need to keep around to send back to Munin) |
| sTitleKW As String | (stoplisted title with only keywords) |
| sBodyKW As String | (stoplisted body with only keywords) |
| sCompany() As String | (companies identified in title or body through pattern matching) |
| sTopic() As String | (topics identified in title or body through pattern matching) |
| sPeople() As String | (people identified in title or body through pattern matching) |
| sWhen() As String | (time identified in title or body through pattern matching) |

-continued

| A.1.1.1.1.1 | Public Type tMeetingRecord |
|---|---|
| sWhere() As String | (location identified in title or body through pattern matching) |
| sLocation As String | (location as passed in by Munin) |
| sTime As String | (time as passed in by Munin) |
| sParticipants() As String | (all participants engaged as passed in by Munin) |
| sMeetingText As String | (the original meeting text w/o userid) |
| EndType | |

There are two other structures which are created to hold each individual pattern utilized in pattern matching. The record tAPatternRecord is an array containing all the components/elements of a pattern. The type tAPatternElement is an array of strings which represent an element in a pattern. Because there may be many "substitutes" for each element, we need an array of strings to keep track of what all the substitutes are. The structures of tAPatternElement and tAPatternRecord are presented below in accordance with a preferred embodiment.

```
Public Type tAPatternElement
  elementArray() As String
End Type
Public Type tAPatternRecord
  patternArray() As tAPatternElement
End Type
```

## Common User Defined Constants

Many constants are defined in each declaration section of the program which may need to be updated periodically as part of the process of maintaining the system in accordance with a preferred embodiment. The constants are accessible to allow dynamic configuration of the system to occur as updates for maintaining the code.

Included in the following tables are lists of constants from each module which I thought are most likely to be modified from time to time. However, there are also other constants used in the code not included in the following list. It does not mean that these non-included constants will never be changed. It means that they will change much less frequently.

For the Main Module (BF.Main):

| CONSTANT | PRESET VALUE | USE |
|---|---|---|
| MSGTOMUNIN_TYPE | 6 | Define the message number used to identify messages between BF and Munin |
| IP_ADDRESS_MUNIN | "10.2.100.48" | Define the IP address of the machine in which Munin and BF are running on so they can transfer data through UDP. |
| PORT_MUNIN | 7777 | Define the remote port in which we are operating on. |
| TIMEOUT_AV | 60 | Define constants for setting time out in inet controls |
| TIMEOUT_NP | 60 | Define constants for setting time out in inet controls |

-continued

For the Main Module (BF.Main):

| CONSTANT | PRESET VALUE | USE |
|---|---|---|
| CMD_SEPARATOR | "\" | Define delimiter to tell which part of Munin's command represents the beginning of our input meeting text |
| OUTPARAM_SEPARATOR | "::" | Define delimiter for separating out different portions of the output. The separator is for delimiting the msg type, the user id, the meeting title and the beginning of the actual stories retrieved. |

For the Search Module (BF.Search):

| CONSTANT | CURRENT VALUE | USE |
|---|---|---|
| PAST_NDAYS | 5 | Define number of days you want to look back for AltaVista articles. Doesn't really matter now because we aren't really doing a news search in alta vista. We want all info. |
| CONNECTOR_AV_URL | "+AND+" | Define how to connect keywords. We want all our keywords in the string so for now use AND. If you want to do an OR or something, just change connector. |
| CONNECTOR_NP_URL | "+AND+" | Define how to connect keywords. We want all our keywords in the string so for now use AND. If you want to do an OR or something, just change connector. |
| NUM_NP_STORIES | 3 | Define the number of stories to return back to Munin from NewsPage. |
| NUM_AV_STORIES | 3 | Define the number of stories to return back to Munin from AltaVista. |

For the Parse Module (BF.Parse):

| CONSTANT | CURRENT VALUE | USE |
|---|---|---|
| PORTION_SEPARATOR | "::" | Define the separator between different portions of the meeting text sent in by Munin. For example in "09::Meet with Chad::about life::Chad\|Denise:::::" "::" is the separator between different parts of the meeting text. |
| PARTICIPANT_SEPARATOR | "\|" | Define the separator between each participant in the participant list portion of the original meeting text. Refer to example above. |

For Pattern Matching Module (BFPatternMatch): There are no constants in this module which require frequent updates.

## General Process Flow

The best way to depict the process flow and the coordination of functions between each other is with the five flowcharts illustrated in FIGS. **2** to **6**. FIG. **2** depicts the overall process flow in accordance with a preferred embodiment. Processing commences at the top of the chart at function block **200** which launches when the program starts. Once the application is started, the command line is parsed to remove the appropriate meeting text to initiate the target of the background find operation in accordance with a preferred embodiment as shown in function block **210**. A global stop list is generated after the target is determined as shown in function block **220**. Then, all the patterns that are utilized for matching operations are generated as illustrated in function block **230**. Then, by tracing through the chart,

function block **200** invokes GoBF **240** which is responsible for logical processing associated with wrapping the correct search query information for the particular target search engine. For example, function block **240** flows to function block **250** and it then calls GoPatternMatch as shown in function block **260**. To see the process flow of GoPatternMatch, we swap to the diagram titled "Process Flow for BF's Pattern Matching Unit."

One key thing to notice is that functions depicted at the same level of the chart are called by in sequential order from left to right (or top to bottom) by their common parent function. For example, Main **200** calls ProcessCommandLinc **210**, then CreateStopListist **220**, then CreatePatterns **230**, then GoBackgroundFinder **240**. FIGS. **3** to **6** detail the logic for the entire program, the parsing unit, the pattern matching unit and the search unit respectively. FIG. **6** details the logic determinative of data flow of key information through BackgroundFinder, and shows the functions that are responsible for creating or processing such information.

## Detailed Search Architecture Under the Simple Query Mode

### Search Alta Vista

(Function Block **270** of FIG. **2**)

The Alta Vista search engine utilizes the identifies and returns general information about topics related to the current meeting as shown in function block **270** of FIG. **2**. The system in accordance with a preferred embodiment takes all the keywords from the title portion of the original meeting text and constructs an advanced query to send to Alta Vista. The keywords are logically combined together in the query. The results are also ranked based on the same set of keywords. One of ordinary skill in the art will readily comprehend that a date restriction or publisher criteria could be facilitated on the articles we want to retrieve. A set of top ranking stories are returned to the calendaring system in accordance with a preferred embodiment.

### News Page

(Function Block **275** of FIG. **2**)

The NewsPage search system is responsible for giving us the latest news topics related to a target meeting. The system takes all of the keywords from the title portion of the original meeting text and constructs a query to send to the NewsPage search engine. The keywords are logically combined together in the query. Only articles published recently are retrieved. The Newspage search system provides a date restriction criteria that is settable by a user according to the user's preference. The top ranking stories are returned to the calendaring system.

FIG. **3** is a user profile data model in accordance with a preferred embodiment. Processing commences at function block **300** which is responsible for invoking the program from the main module. Then, at function block **310**, a wrapper function is invoked to prepare for the keyword extraction processing in function block **320**. After the keywords are extracted, then processing flows to function block **330** to determine if the delimiters are properly positioned. Then, at function block **340**, the number of words in a particular string is calculated and the delimiters for the particular field are and a particular field from the meeting text is retrieved at function block **350**. Then, at function block **380**, the delimiters of the string are again checked to assure they are placed appropriately. Finally, at function block **360**, the extraction of each word from the title and body of the message is performed a word at a time utilizing the logic in function block **362** which finds the next closest word delimiter in the input phrase, function block **364** which strips unnecessary materials from a word and function block **366** which determines if a word is on the stop list and returns an error if the word is on the stop list.

## Pattern Matching in Accordance with a Preferred Embodiment

The limitations associated with a simple searching method include the following:

1. Because it relies on a stoplist of unwanted words in order to extract from the meeting text a set of keywords, it is limited by how comprehensive the stoplist is. Instead of trying to figure out what parts of the meeting text we should throw away, we should focus on what parts of the meeting text we want.
2. A simple search method in accordance with a preferred embodiment only uses the keywords from a meeting title to form queries to send to Alta Vista and NewsPage. This ignores an alternative source of information for the query, the body of the meeting notice. We cannot include the keywords from the meeting body to form our queries because this often results in queries which are too long and so complex that we often obtain no meaningful results.
3. There is no way for us to tell what each keyword represents. For example, we may extract "Andy" and "Grove" as two keywords. However, a simplistic search has no way knowing that "Andy Grove" is in fact a person's name. Imagine the possibilities if we could somehow intelligently guess that "Andy Grove" is a person's name. Information such as where he is employed and currently resides.
4. In summary, by relying solely on a stoplist to parse out unnecessary words, we suffer from "information overload".

## Pattern Matching Overcomes these Limitations in Accordance with a Preferred Embodiment

Here is how the pattern matching system can address each of the corresponding issues above in accordance with a preferred embodiment.

1. By doing pattern matching, we match up only parts of the meeting text that we want and extract those parts.
2. By performing pattern matching on the meeting body and extracting only the parts from the meeting body that we want. Our meeting body will not go to complete waste then.
3. Pattern matching is based on a set of templates that we specify, allowing us to identify people names, company names and other items from a meeting text.
4. In summary, with pattern matching, we no longer suffer from information overload. Of course, the big problem is how well our pattern matching works. If we rely exclusively on artificial intelligence processing, we do not have a 100% hit rate. We are able to identify about 20% of all company names presented to us.

### Patterns

A pattern in the context of a preferred embodiment is a template specifying the structure of a phrase we are looking

for in a meeting text. The patterns supported by a preferred embodiment are selected because they are templates of phrases which have a high probability of appearing in someone's meeting text. For example, when entering a meeting in a calendar, many would write something such as "Meet with Bob Dutton from Stanford University next Tuesday." A common pattern would then be something like the word "with" followed by a person's name (in this example it is Bob Dutton) followed by the word "from" and ending with an organization's name (in this case, it is Stanford University).

Pattern Matching Terminology

The common terminology associated with pattern matching is provided below.

Pattern: a pattern is a template specifying the structure of a phrase we want to bind the meeting text to. It contains sub units.

Element: a pattern can contain many sub-units. These subunits are called elements. For example, in the pattern "with $PEOPLE$ from $COMPANY$", "with" "$PEOPLE$" "from" "$COMPANY$" are all elements.

Placeholder: a placeholder is a special kind of element in which we want to bind a value to.Using the above example, "$PEOPLE$" is a placeholder.

Indicator: an indicator is another kind of element which we want to find in a meeting text but no value needs to bind to it. There may be often more than one indicator we are looking for in a certain pattern. That is why an indicator is not an "atomic" type.

Substitute: substitutes are a set of indicators which are all synonyms of each other. Finding any one of them in the input is good.

There are five fields which are identified for each meeting:

| | | |
|---|---|---|
| ◆ | Company | ($COMPANY$) |
| ◆ | People | ($PEOPLE$) |
| ◆ | Location | ($LOCATION$) |
| ◆ | Time | ($TIME$) |
| ◆ | Topic | ($TOPIC_UPPER$) or ($TOPIC_ALL$) |

In parentheses are the placeholders I used in my code as representation of the corresponding meeting fields.

Each placeholder has the following meaning:

$COMPANY$: binds a string of capitalized words (e.g., Meet with Joe Carter of <Andersen Consulting >)

$PEOPLE$: binds series of string of two capitalized words potentially connected by "," "and" or "&" (e.g., Meet with <Joe Carter> of Andersen Consulting, Meet with <Joe Carter and Luke Hughes> of Andersen Consulting)

$LOCATION$: binds a string of capitalized words (e.g., Meet Susan at <Palo Alto Square>)

$TIME$: binds a string containing the format #:## (e.g., Dinner at <6:30 pm>)

$TOPIC_UPPER$: binds a string of capitalized words for our topic (e.g., <Stanford Engineering Recruiting> Meeting to talk about new hires).

$TOPIC_ALL$: binds a string of words without really caring if it's capitalized or not. (e.g., Meet to talk about <ubiquitous computing>)

Here is a table representing all the patterns supported by BF. Each pattern belongs to a pattern group. All patterns within a pattern group share a similar format and they only differ from each other in terms of what indicators are used as substitutes. Note that the patterns which are grayed out are also commented in the code. BF has the capability to support these patterns but we decided that matching these patterns is not essential at this point.

| PAT GRP | PAT # | PATTERN | EXAMPLE |
|---|---|---|---|
| 1 | a | $PEOPLE$ of $COMPANY$ | Paul Maritz of Microsoft |
| | b | $PEOPLE$ from $COMPANY$ | Bill Gates, Paul Allen and Paul Maritz from Microsoft |
| 2 | a | $TOPIC_UPPER$ meeting | Push Technology Meeting |
| | b | $TOPIC_UPPER$ mtg | Push Technology Mtg |
| | c | $TOPIC_UPPER$ demo | Push Technology demo |
| | d | $TOPIC_UPPER$ interview | Push Technology interview |
| | e | $TOPIC_UPPER$ presentation | Push Technology presentation |
| | f | $TOPIC_UPPER$ visit | Push Technology visit |
| | g | $TOPIC_UPPER$ briefing | Push Technology briefing |
| | h | $TOPIC_UPPER$ discussion | Push Technology discussion |
| | i | $TOPIC_UPPER$ workshop | Push Technology workshop |
| | j | $TOPIC_UPPER$ prep | Push Technology prep |
| | k | $TOPIC_UPPER$ review | Push Technology review |
| | l | $TOPIC_UPPER$ lunch | Push Technology lunch |
| | m | $TOPIC_UPPER$ project | Push Technology project |
| | n | $TOPIC_UPPER$ projects | Push Technology projects |
| 3 | a | $COMPANY$ corporation | Intel Corporation |
| | b | $COMPANY$ corp. | IBM Corp. |
| | c | $COMPANY$ systems | Cisco Systems |
| | d | $COMPANY$ limited | IBM limited |
| | e | $COMPANY$ ltd | IBM ltd |
| 4 | a | about $TOPIC_ALL$ | About intelligent agents technology |
| | b | discuss $TOPIC_ALL$ | Discuss intelligent agents technology |
| | c | show $TOPIC_ALL$ | Show the client our intelligent agents technology |
| | d | re: $TOPIC_ALL$ | re: intelligent agents technology |
| | e | review $TOPIC_ALL$ | Review intelligent agents technology |
| | f | agenda | The agenda is as follows:<br>—clean up<br>—clean up<br>—clean up |
| | g | agenda: $TOPIC_ALL$ | Agenda:<br>—demo client intelligent agents technology.<br>—demo ecommerce. |
| 5 | a | w/$PEOPLE$ of $COMPANY$ | Meet w/Joe Carter of Andersen Consulting |
| | b | w/$PEOPLE$ from $COMPANY$ | Meet w/Joe Carter from Andersen Consulting |
| 6 | a | w/$COMPANY$ per $PEOPLE$ | Talk w/Intel per Jason Foster |
| 7 | a | At $TIME$ | at 3:00 pm |
| | b | Around $TIME$ | Around 3:00 pm |
| 8 | a | At $LOCATION$ | At LuLu's restaurant |
| | b | In $LOCATION$ | in Santa Clara |
| 9 | a | Per $PEOPLE$ | per Susan Butler |
| 10 | a | call w/$PEOPLE$ | Conf call w/John Smith |
| | B | call with $PEOPLE$ | Conf call with John Smith |
| 11 | A | prep for $TOPIC_ALL$ | Prep for London meeting |
| | B | preparation for $TOPIC_ALL$ | Preparation for London meeting |

FIG. 4 is a detailed flowchart of pattern matching in accordance with a preferred embodiment. Processing commences at function block **400** where the main program invokes the pattern matching application and passes control to function block **410** to commence the pattern match

processing. Then, at function block **420**, the wrapper function loops through to process each pattern which includes determining if a part of the text string can be bound to a pattern as shown in function block **430**. Then, at function block **440**, various placeholders are bound to values if they exist, and in function block **441**, a list of names separated by punctuation are bound, and at function block **442** a full name is processed by finding two capitalized words as a full name and grabbing the next letter after a space after a word to determine if it is capitalized. Then, at function block **443**, time is parsed out of the string in an appropriate manner and the next word after a blank space in function block **444**. Then, at function block **445**, the continuous phrases of capitalized words such as company, topic or location are bound and in function block **446**, the next word after the blank is obtained for further processing in accordance with a preferred embodiment. Following the match meeting field processing, function block **450** is utilized to locate an indicator which is the head of a pattern, the next word after the blank is obtained as shown in function block **452** and the word is checked to determine if the word is an indicator as shown in function block **454**. Then, at function block **460**, the string is parsed to locate an indicator which is not at the end of the pattern and the next word after unnecessary white space such as that following a line feed or a carriage return is processed as shown in function block **462** and the word is analyzed to determine if it is an indicator as shown in function block **464**. Then, in function block **470**, the temporary record is reset to the null set to prepare it for processing the next string and at function block **480**, the meeting record is updated and at function block **482** a check is performed to determine if an entry is already made to the meeting record before parsing the meeting record again.

### Using the Identified Meeting Fields

Now that we have identified fields within the meeting text which we consider important, there are quite a few things we can do with it. One of the most important applications of pattern matching is of course to improve the query we construct which eventually gets submitted to Alta Vista and News Page. There are also a lot of other options and enhancements which exploit the results of pattern matching that we can add to BF. These other options will be described in the next section. The goal of this section is to give the reader a good sense of how the results obtained from pattern matching can be used to help us obtain better search results.

FIG. **5** is a flowchart of the detailed processing for preparing a query and obtaining information from the Internet in accordance with a preferred embodiment. Processing commences at function block **500** and immediately flows to function block **510** to process the wrapper functionality to prepare for an Internet search utilizing a web search engine. If the search is to utilize the Alta Vista search engine, then at function block **530**, the system takes information from the meeting record and forms a query in function blocks **540** to **560** for submittal to the search engine. If the search is to utilize the NewsPage search engine, then at function block **520**, the system takes information from the meeting record and forms a query in function blocks **521** to **528**.

### Alta Vista Search Engine

The strength of the Alta Vista search engine is that it provides enhanced flexibility. Using its advance query method, one can construct all sorts of Boolean queries and rank the search however you want. However, one of the biggest drawbacks with Alta Vista is that it is not very good at handling a large query and is likely to give back irrelevant results. If we can identify the topic and the company within a meeting text, we can form a pretty short but comprehensive query which will hopefully yield better results. We also want to focus on the topics found. It may not be of much merit to the user to find out info about a company especially if the user already knows the company well and has had numerous meetings with them. It's the topics they want to research on.

### News Page Search Engine

The strength of the News Page search engine is that it does a great job searching for the most recent news if you are able to give it a valid company name. Therefore when we submit a query to the news page web site, we send whatever company name we can identify and only if we cannot find one do we use the topics found to form a query. If neither one is found, then no search is performed. The algorithmn utilized to form the query to submit to Alta Vista is illustrated in FIG. **7**. The algorithmn that we will use to form the query to submit to News Page is illustrated in FIG. **8**.

The following table describes in detail each function in accordance with a preferred embodiment. The order in which functions appear mimics the process flow as closely as possible. When there are situations in which a function is called several times, this function will be listed after the first function which calls it and its description is not duplicated after every subsequent function which calls it.

| Procedure Name | Type | Called By | Description |
|---|---|---|---|
| Main (BF.Main) | Public Sub | None | This is the main function where the program first launches. It initializes BF with the appropriate parameters (e.g., Internet time-out, stoplist. . .) and calls GoBF to launch the main part of the program. |
| ProcessCommandLine (BF.Main) | Private Sub | Main | This function parses the command line. It assumes that the delimiter indicating the beginning of imput from Munin is stored in the constant CMD_SEPARATOR. |
| CreateStopList (BF.Main) | Private Function | Main | This function sets up a stop list for future use to parse out unwanted words from the meeting text. There are commas on each side of each word to enable straight checking. |
| CreatePatterns (BF.Pattern Match) | Public Sub | Main | This procedure is called once when BF is first initialized to create all the potential patterns that portions of the meeting text can bind to. A pattern can contain however many elements as needed. There are two types of elements. The first type of elements are indicators. These are real words which delimit the potential of a meeting field (eg company) to follow. Most of these indicators are stop words as expected because stop words are words usually common to all meeting text so it makes sense they form patterns. The second type of elements are special strings which represent |

-continued

| Procedure Name | Type | Called By | Description |
|---|---|---|---|
| | | | placeholders. A placeholder is always in the form of $*$ where * can be either PEOPLE, COMPANY, TOPIC_UPPER, TIME, LOCATION or TOPIC_ALL. A pattern can begin with either one of the two types of elements and can be however long, involving however any number/type of elements. This procedure dynamically creates a new pattern record for each pattern in the table and it also dynamically creates new tAPatternElements for each element within a pattern. In addition, there is the concept of being able to substitute indicators within a pattern. For example, the pattern $PEOPLE$ of $COMPANY$ is similar to the pattern $PEOPLE$ from $COMPANY$. "from" is a substitute for "of". Our structure should be able to express such a need for substitution. |
| GoBF (BF.Main) | Public Sub | Main | This is a wrapper procedure that calls both the parsing and the searching subroutines of the BF. It is also responsible for sending data back to Munin. |
| ParseMeeting Text (BF.Parse) | Public Function | GoBackGroundFinder | This function takes the initial meeting text and identifies the userID of the record as well as other parts of the meeting text including the title, body, participant list, location and time. In addition, we call a helper function ProcessStopList to eliminate all the unwanted words from the originalmeeting title and meeting body so that only keywords are left. The information parsed out is stored in the MeetingRecord structure. Note that this function does no error checking and for the most time assumes that the meeting text string is correctly formatted by Munin. The important variable is thisMeeting Record is the temp holder for all info regarding current meeting. It's eventually returned to caller. |
| FormatDelimitation (BF.Parse) | Private | ParseMeetingText, DetermineNum Words, GetAWordFrom String | There are 4 ways in which the delimiters can be placed. We take care of all these cases by reducing them down to Case 4 in which there are no delimiters around but only between fields in a string(e.g., A::B::C) |
| DetermineNumWords (BF.Parse) | Public Function | ParseMeeting Text, ProcessStop List | This functions determines how many words there are in a string (stInEvalString) The function assumes that each word is separated by a designated separator as specified in stSeparator. The return type is an integer that indicates how many words have been found assuming each word in the string is separated by stSeparator. This function is used along with GetAWordFromString and should be called before calling GetAWordFrom String. |
| GetAWordFromString (BF.Parse) | Public Function | ParseMeeting Text, ProcessStop List | This function extracts the ith word of the string(stInEvalString)assuming that each word in the string is separated by a designated separator contained in the variable stSeparator. In most cases, use this function with DetermineNumWords. The function returns the wanted word. This function checks to make sure that iInWordNum is within bounds so that i is not greater than the total number of words in string or less than/equal to zero. If it is out of bounds, we return empty string to indicate we can't get anything. We try to make sure this doesn't happen by calling DetermineNumWords first. |
| ParseAndCleanPhrase (BF.Parse) | Private Function | ParseMeetingText | This function first grabs the word and send it to CleanWord in order strip the stuff that nobody wants. There are things in parseWord that will kill the word, so we will need a method of looping through the body and rejecting words without killing the whole function i guess keep CleanWord and check a return value ok, now I have a word so I need to send it down the parse chain. This chain goes ParseCleanPhrase -> CleanWord -> EvaluateWord. If the word gets through the entire chain without being killed, it will be added at the end to our keyword string. first would be the function that checks for "/" as a delimiter and extracts the parts of that. This I will call "StitchFace" (Denise is more normal and calls it GetAWordFromString) if this finds words, then each of these will be sent, in turn, down the chain. If these get through the entire chain without being added or killed then they will be added rather than tossed. |
| FindMin (BF.Parse) | Private Function | ParseAndCleanPhrase | This function takes in 6 input values and evaluates to see what the minimum non zero value is. It first creates an array as a holder so that can sort the five input values in ascending order. Thus the minimum value will be the first non zero value element of the array. If we go through entire array without finding a non zero value, we know that there is an error and we exit the function. |
| CleanWord (BF.Parse) | Private Function | ParseAndCleanPhrase | This function tries to clean up a word in a meeting text. It first of all determines if the string is of a valid length. It then passes it through a series of tests to see it is and when needed, it will edit the word and strip unnecessary characters off of it. Such tests includes getting rid of file extensions, non chars, numbers etc. |
| EvaluateWord (BF.Parse) | Private Function | ParseAndCleanPhrase | This function tests to see if this word is in the stop list so it can determine whether to eliminate the word from the original meeting text. If a word is not in the stoplist, it should stay around as a keyword and this function exits beautifully with no errors. However, if the words is a stopword, an error must be returned. We must properly delimit the input test string so we don't accidentally retrieve substrings. |
| GoPatternMatch (BF.Pattern Match) | Public Sub | GoBF | This procedure is called when our QueryMethod is set to complex query meaning we do want to do all the pattern matching stuff. It's a simple wrapper function which initializes some arrays and then invokes pattern matching on the title and the body. |
| MatchPatterns (BF.Pattern Match) | Public Sub | GoPattern Match | This procedure loops through every pattern in the pattern table and tries to identify different fields within a meeting text specified by sInEvalString. For debugging purposes it also tries to tabulate how many times a certain pattern was triggered and stores it in gTabulateMatches to see which pattern fired the most. gTabulateMatches is stored as a global because we want to be able to run a batch file of 40 or 50 test strings and still be able to know how often a pattern was triggered. |
| MatchAPattern (BF.Pattern | Private Function | MatchPatterns | This function goes through each element in the current pattern. It first evaluates to determine whether element is a placeholder or an indicator. If it is a placeholder, then it |

-continued

| Procedure Name | Type | Called By | Description |
|---|---|---|---|
| Match) | | | will try to bind the placeholder with some value. If it is an indicator, then we try to locate it. There is a trick however. Depending on whether we are at current element is the head of the pattern or not we want to take different actions. If we are at the head, we want to look for the indicator or the placeholder. If we can't find it, then we know that the current pattern doesn't exist and we quit. However, if it is not the head, then we continue looking, because there may still be a head somewhere. We retry in this case. |
| etingField (BF.Pattern Match) | Private Function | MatchAPattern | This function uses a big switch statement to first determine what kind of placeholder we are talking about and depending on what type of placeholder, we have specific requirements and different binding criteria as specified in the subsequent functions called such as BindNames, BindTime etc. If binding is successful we add it to our guessing record. |
| BindNames (BF.Pattern Match) | Private Function | MatchMeetingField | In this function, we try to match names to the corresponding placeholder $PEOPLE$. Names are defined as any consecutive two words which are capitalized. We also what to retrieve a series of names which are connected by and, or & so we look until we don't see any of these 3 separators anymore. Note that we don't want to bind single word names because it is probably too general anyway so we don't want to produce broad but irrelevant results. This function calls BindAFullName which binds one name so in a since BindNames collects all the results from BindAFullName |
| BindAFullName (BF.Pattern Match) | Private Function | BindNames | This function tries to bind a full name. If the $PEOPLE$ placeholder is not the head of the pattern, we know that it has to come right at the beginning of the test string because we've been deleting stuff off the head of the string all along. If it is the head, we search until we find something that looks like a full name. If we can't find it, then there's no such pattern in the text entirely and we quit entirely from this pattern. This should eventually return us to the next pattern in MatchPatterns. |
| GetNextWordAfter-WhiteSpace (BF.Pattern Match) | Private Function | BindAFull Name, BindTime, BindCompanyTo picLoc | This function grabs the next word in a test string. It looks for the next word after white spaces, @ or/. The word is defined to end when we encounter another one of these white spaces or separators. |
| BindTime (BF.Pattern Match) | Private Function | MatchMeetingField | Get the immediate next word and see if it looks like a time pattern. If so we've found a time and so we want to add it to the record. We probably should add more time patterns But people don't seem to like to enter the time in their titles these days especially since we now have tools like OutLook |
| BindCompany-TopicLoc (BF.Pattern Match) | Private Function | MatchMeetingField | This function finds a continuous capitalized string and binds it to stMatch which is passed by reference from MatchMeetingField. A continuous capitalized string is a sequence of capitalized words which are not interrupted by things like, etc. There's probably more stuff we can add to the list of interruptions. |
| LocatePatternHead (BF.Pattern Match) | Private Function | MatchAPattern | This function tries to locate an element which is an indicator. Note that this indicator SHOULD BE AT THE HEAD of the pattern otherwise it would have gone to the function LocateIndicator instead. Therefore, we keep on grabbing the next word until either there's no word for us to grab (quit) or if we find one of the indicators we are looking for. |
| ContainInArray (BF.Pattern Match) | Private Function | LocatePattern Head, LocateIndicator | 'This function is really simple. It loops through all the elements in the array 'to find a matching string. |
| LocateIndicator (BF.Pattern Match) | Private Function | MatchAPattern | This function tries to locate an element which is an indicator. Note that this indicator is NOT at the head of the pattern otherwise it would have gone to LocatePatternHead instead. Because of this, if our pattern is to be satisfied, the next word we grab HAS to be the indicator or else we would have failed. Thus we only grab one word, test to see if it is a valid indicator and then return result. |
| InitializeGuess-Record (BF.Pattern Match) | Private Sub | MatchAPattern | This function reinitializes our temporary test structure because we have already transferred the info to the permanent structure, we can reinitialize it so they each have one element |
| AddToMeeting-Record (BF.Pattern | Private Sub | MatchAPattern | This function is only called when we know that the information stored in tInCurrGuesses is valid meaning that it represents legitimate guesses of meeting fields ready to be stored in the permanent Match) record, tInMeetingRecord. We check to make sure that we do not store duplicates and we also what to clean up what we want to store so that there's no clutter such as punctuation, etc. The reason why we don't clean up until now is to save time. We don't waste resources calling ParseAndCleanPhrase until we know for sure that we are going to add it permanently. |
| NoDuplicate Entry (BF.Pattern Match) | Private Function | AddToMeetingRecord | This function loops through each element in the array to make sure that the test string aString is not the same as any of the strings already stored in the array. Slightly different from ContainInArray. |
| SearchAltaVista (BF.Search) | Public Function | GoBackGroundFinder | This function prepares a query to be submitted to AltaVista Search engine. It submits it and then parses the returning result in the appropriate format containing the title, URL and body/summary of each story retrieved. The number of stories retrieved is specified by the constant NUM_AV_ STORIES. Important variables include stURLAltaVista used to store query to submit stResultHTML used to store html from page specified by stURLAltaVista. |
| ConstructAlta-VistaURL (BF.Search) | Private Function | SearchAltaVista | This function constructs the URL string for the alta vista search engine using the advanced query search mode. It includes the keywords to be used, the language and how we want to rank the search. Depending on whether we want to use the results of |

-continued

| Procedure Name | Type | Called By | Description |
|---|---|---|---|
| | | | our pattern matching unit, we construct our query differently. |
| ConstructSimple KeyWord (BF.Search) | Private Function | ConstructAltaVistaURL ConstructNewsPageURL | This function marches down the list of keywords stored in the stTitleKW or stBodyKW fields of the input meeting record and links them up into one string with each keyword separated by a connector as determined by the input variable stInConnector. Returns this newly constructed string. |
| ConstructComplex-AVKeyWord (BF.Search) | Private Function | ConstructAltaVistaURL | This function constructs the keywords to be send to the AltaVista site. Unlike ConstructSimpleKeyWord which simply takes all the keywords from the title to form the query, this function will look at the results of BF's pattern matching process and see if we are able to identify any specific company names or topics for constructing the queries. Query will include company and topic identified and default to simple query if we cannot identify either company or topic. |
| JoinWithConnectors (BF.Search) | Private Function | ConstructComplex-AVKeyWord, ConstructComplex-NPKeyWord, RefineWithRank | This function simply replaces the spaces between the words within the string with a connector which is specified by the input. |
| RefineWithDate (NOT CALLED AT THE MOMENT) (BF.Search) | Private Function | ConstructAltaVistaURL | This function constructs the date portion of the alta vista query and returns this portion of the URL as a string. It makes sure that alta vista searches for articles within the past PAST_NDAYS. |
| RefineWithRank (BF.Search) | Private Function | ConstructAltaVistaURL | This function constructs the string needed to passed to Altavista in order to rank an advanced query search. If we are constructing the simple query we will take in all the keywords from the title. For the complex query, we will take in words from company and topic, much the same way we formed the query in ConstructComplexAVKeyWord |
| IdentifyBlock (BF.Parse) | Public Function | SearchAltaVista, SearchNewsPage | This function extracts the block within a string marked by the beginning and the ending tag given as inputs starting at a certain location (iStart). The block retrieved does not include the tags themselves. If the block cannot be identified with the specified delimiters, we return unsuccessful through the parameter iReturnSuccess passed to use by reference. The return type is the the block retrieved. |
| IsOpenURLError (BF.Error) | Public Function | SearchAltaVista, SearchNewsPage | This function determines whether the error encountered is that of a timeout error. It restores the mouse to default arrow and then returns true if it is a time out or false otherwise. |
| SearchNews Page (BF.Search) | Public Function | GoBackGroundFinder | This function prepares a query to be submitted to NewsPage Search engine. It submits it and then parses the returning result in the appropriate format containing the title, URL and body/summary of each story retrieved. The number of stories retrieved is specified by the constant UM_NP_STORIES |
| ConstructNews-PageURL (BF.Search) | Private Function | SearchNewsPage | This function constructs the URL to send to the NewsPage site. It uses the information contained in the input meeting record to determine what keywords to use. Also depending whether we want simple or complex query, we call different functions to form strings. |
| ConstructComplex-NPKeyWord (BF.Search) | Private Function | ConstructNewsPageURL | This function constructs the keywords to be send to the NewsPage site. Unlike ConstructKeyWordString which simply takes all the keywords from the title to form the query, this function will look at the results of BF's pattern matching process and see if we are able to identify any specific company names or topics for constructing the queries. Since newspage works best when we have a company name, we'll use only the company name and only if there is no company will we use topic. |
| ConstructOverall-Result (BF.Main) | Private Function | GoBackGroundFinder | This function takes in as input an array of strings (stInStories) and a MeetingRecord which stores the information for the current meeting. Each element in the array stores the stories retrieved from each information source. The function simply constructs the appropriate output to send to Munin including a return message type to let Munin know that it is the BF responding and also the original user_id and meeting title so Munin knows which meeting BF is talking about. |
| ConnectAnd TransferTo Munin (BF.Main) | Public Sub | GoBackGroundFinder | This function allows Background Finder to connect to Munin and eventually transport information to Munin. We will be using the UDP protocol instead of the TCP protocol so we have to set up the remote host and port correctly. We use a global string to store gResult Overall because although it is unnecessary with UDP, it is needed with TCP and if we ever switch back don't want to change code. |
| DisconnectFrom-MuninAndQuit (BF.Main) | Public Sub | | |

FIG. **6** is a flowchart of the actual code utilized to prepare and submit searches to the Alta Vista and Newspage search engines in accordance with a preferred embodiment. Processing commences at function block **610** where a command line is utilized to update a calendar entry with specific calendar information. The message is next posted in accordance with function block **620** and a meeting record is created to store the current meeting information in accordance with function block **630**. Then, in function block **640** the query is submitted to the Alta Vista search engine and in function block **650**, the query is submitted to the Newspage search engine. When a message is returned from the search engine, it is stored in a results data structure as shown in function block **660** and the information is processed and stored in summary form in a file for use in preparation for the meeting as detailed in function block **670**.

FIG. **7** provides more detail on creating the query in accordance with a preferred embodiment. Processing commences at function block **710** where the meeting record is parsed to obtain potential companies, people, topics, loca-

tion and a time. Then, in function block **720**, at least one topic is identified and in function block **720**, at least one company name is identified and finally in function block **740**, a decision is made on what material to transmit to the file for ultimate consumption by the user.

FIG. **8** is a variation on the query theme presented in FIG. **7**. A meeting record is parsed in function block **800**, a company is identified in function block **820**, a topic is identified in function block **830** and finally in function block **840** the topic and or the company is utilized in formulating the query.

Alternative embodiments for adding various specific features for specific user requirements are discussed below.

Enhance Target Rate for Pattern Matching

To increase BF's performance, more patterns/pattern groups are added to the procedure "CreatePatterns." The existing code for declaring patterns can be used as a template for future patterns. Because everything is stored as dynamic arrays, it is convenient to reuse code by cutting and pasting. The functions BindName, BindTime, BindCompanyLocTopic which are responsible for associating a value with a placeholder can be enhanced. The enhancement is realized by increasing the set of criteria for binding a certain meeting field in order to increase the number of binding values. For example, BindTime currently accepts and binds all values in the form of ##:## or #:##. To increase the times we can bind, we may want BindTime to also accept the numbers **1** to **12** followed by the more aesthetic time terminology "o'clock," Vocabulary based recognition algorithms and assigning an accuracy rate to each guess BF makes allowing only guesses which meet a certain threshold to be valid.

Depending on what location the system identifies through pattern matching or alternatively depending on what location the user indicates as the meeting place, a system in accordance with a preferred embodiment suggests a plurality of fine restaurants whenever it detects the words lunch/dinner/breakfast. We can also use a site like company finder to confirm what we got is indeed a company name or if there is no company name that pattern matching can identify, we can use a company finder web site as a "dictionary" for us to determine whether certain capitalized words represent a company name. We can even display stock prices and breaking news for a company that we have identified.

Wireless Bargain Identification in Accordance with a Preferred Embodiment

FIG. **9** is a flow diagram that depicts the hardware and logical flow of control for a device and a software system designed to allow Web-based comparison shopping in conventional, physical, non-Web retail environments. A wireless phone or similar hand-held wireless device **920** with Internet Protocol capability is combined with a miniature barcode reader **910** (installed either inside the phone or on a short cable) and used to scan the Universal Product Code (UPC) bar code on a book or other product **900**. The wireless device **920** transmits the bar code via an antennae **930** to the Pocket BargainFinder Service Module (running on a Web server) **940**, which converts it to (in the case of books) its International Standard Book Number or (in the case of other products) whatever identifier is appropriate. The Service Module then contacts the appropriate third-party Web site(s) to find price, shipping and availability information on the product from various Web suppliers **950**. This information is formatted and displayed on the hand-held device's screen. The IP wireless phone or other hand held device **920** utilizes a wireless modem such as a Ricochet SE Wireless Modem from Metricom. Utilizing this device, a user can hang out in a coffee shop with a portable computer perched on a rickety little table, with a latte sloshing dangerously close to the keyboard, and access the Internet at speeds rivaling direct connect via a telephone line.

The 8-ounce Ricochet SE Wireless Modem is about as large as a pack of cigarettes and setup is extremely simple, simply attach the modem to the back of your portable's screen with the included piece of Velcro, plug the cable into the serial port, flip up the stubby antenna, and transmit. Software setup is equally easy: a straightforward installer adds the Ricochet modem drivers and places the connection icon on your desktop. The functional aspects of the modem are identical to that of a traditional telephone modem.

Of course, wireless performance isn't nearly as reliable as a traditional dial-up phone connection. We were able to get strong connections in several San Francisco locations as long as we stayed near the windows. But inside CNET's all-brick headquarters, the Ricochet couldn't connect at all. When you do get online, performance of up to 28.8 kbps is available with graceful degradation to slower speeds. But even the slower speeds didn't disappoint. Compared to the alternative—connecting via a cellular modem—the Ricochet is much faster, more reliable, and less expensive to use. Naturally, the SE Wireless is battery powered. The modem has continuous battery life of up to 12 hours. And in accordance with a preferred embodiment, we ran down our portable computer's dual cells before the Ricochet started to fade.

Thus, utilizing the wireless modem, a user may utilize the web server software **940** to identify the right product **950** and then use an appropriate device's key(s) to select a supplier and place an order in accordance with a preferred embodiment. The BargainFinder Service Module then consummates the order with the appropriate third-party Web supplier **960**.

mySite! Personal Web Site & Intentions Value Network Prototype

mySite! is a high-impact, Internet-based application in accordance with a preferred embodiment that is focused on the theme of delivering services and providing a personalized experience for each customer via a personal web site in a buyer-centric world. The services are intuitively organized around satisfying customer intentions—fundamental life needs or objectives that require extensive planning decisions, and coordination across several dimensions, such as financial planning, healthcare, personal and professional development, family life, and other concerns. Each member owns and maintains his own profile, enabling him to create and browse content in the system targeted specifically at him. From the time a demand for products or services is entered, to the completion of payment, intelligent agents are utilized to conduct research, execute transactions and provide advice. By using advanced profiling and filtering, the intelligent agents learn about the user, improving the services they deliver. Customer intentions include Managing Daily Logistics (e.g., email, calendar, contacts, to-do list, bill payment, shopping, and travel planning); and Moving to a New Community (e.g., finding a place to live, moving household possessions, getting travel and shipping insurance coverage, notifying business and personal contacts, learning about the new community). From a consumer standpoint, mySite! provides a central location where a user can access

relevant products and services and accomplish daily tasks with ultimate ease and convenience.

From a business standpoint, mySite! represents a value-added and innovative way to effectively attract, service, and retain customers. Intention value networks allow a user to enter through a personalized site and, and with the assistance of a learning, intelligent agent, seamlessly interact with network participants. An intention value network in accordance with a preferred embodiment provides superior value. It provides twenty four hour a day, seven days a week access to customized information, advice and products. The information is personalized so that each member views content that is highly customized to assure relevance to the required target user.

Egocentric Interface

An Egocentric Interface is a user interface crafted to satisfy a particular user's needs, preferences and current context. It utilizes the user's personal information that is stored in a central profile database to customize the interface. The user can set security permissions on and preferences for interface elements and content. The content integrated into the Egocentric Interface is customized with related information about the user. When displaying content, the Egocentric Interface will include the relationship between that content and the user in a way that demonstrates how the content relates to the user. For instance, when displaying information about an upcoming ski trip the user has signed up for, the interface will include information about events from the user's personal calendar and contact list, such as other people who will be in the area during the ski trip. This serves to put the new piece of information into a context familiar to the individual user.

FIG. **10**A describes the Intention Value Network Architecture implementation for the World Wide Web. For simplification purposes, this diagram ignores the complexity pertaining to security, scalability and privacy. The customer can access the Intention Value Network with any Internet web browser **1010**, such as Netscape Navigator or Microsoft Internet Explorer, running on a personal computer connected to the Internet or a Personal Digital Assistant with wireless capability. See FIG. **17** for a more detailed description of the multiple methods for accessing an Intention value Network. The customer accesses the Intention Value Network through the unique name or IP address associated with the Integrator's Web Server **1020**. The Integrator creates the Intention Value Network using a combination of resources, such as the Intention Database **1030**, the Content Database **1040**, the Supplier Profile Database **1050**, and the Customer Profile Database **1060**.

The Intention Database **1030** stores all of the information about the structure of the intention and the types of products and services needed to fulfill the intention. Information in this database includes intention steps, areas of interest, layout templates and personalization templates. The Content Database **1040** stores all of the information related to the intention, such as advice, referral information, personalized content, satisfaction ratings, product ratings and progress reports.

The Supplier Profile Database **1050** contains information about the product and service providers integrated into the intention. The information contained in this database provides a link between the intention framework and the suppliers. It includes product lists, features and descriptions, and addresses of the suppliers' product web sites. The Customer Profile Database **1060** contains personal information about the customers, such as name, address, social security number and credit card information, personal preferences, behavioral information, history, and web site layout preferences. The Supplier's Web Server **1070** provides access to all of the supplier's databases necessary to provide information and transactional support to the customer.

The Product Information Database **1080** stores all product-related information, such as features, availability and pricing. The Product Order Database **1090** stores all customer orders. The interface to this database may be through an Enterprise Resource Planning application offered by SAP, Baan, Oracle or others, or it may be accessible directly through the Supplier's Web Server or application server. The Customer Information Database **1091** stores all of the customer information that the supplier needs to complete a transaction or maintain customer records.

FIG. **10**B is a flowchart providing the logic utilized to create a web page within the Egocentric Interface. The environment assumes a web server and a web browser connected through a TCP/IP network, such as over the public Internet or a private Intranet. Possible web servers could include Microsoft Internet Information Server, Netscape Enterprise Server or Apache. Possible web browsers include Microsoft Internet Explorer or Netscape Navigator. The client (i.e. web browser) makes a request **1001** to the server (i.e. web server) for a particular web page. This is usually accomplished by a user clicking on a button or a link within a web page. The web server gets the layout and content preferences **1002** for that particular user, with the request to the database keyed off of a unique user id stored in the client (i.e. web browser) and the User profile database **1003**. The web server then retrieves the content **1004** for the page that has been requested from the content database **1005**. The relevant user-centric content, such as calendar, email, contact list, and task list items are then retrieved **1006**. (See FIG. **11** for a more detailed description of this process.) The query to the database utilizes the user content preferences stored as part of the user profile in the User profile database **1003** to filter the content that is returned. The content that is returned is then formatted into a web page **1007** according to the layout preferences defined in the user profile. The web page is then returned to the client and displayed to the user **1008**.

FIG. **11** describes the process of retrieving user-centric content to add to a web page. This process describes **1006** in FIG. **10**B in a more detailed fashion. It assumes that the server already has obtained the user profile and the existing content that is going to be integrated into this page. The server parses **1110** the filtered content, looking for instances of events, contact names and email addresses. If any of these are found, they are tagged and stored in a temporary holding space. Then, the server tries to find any user-centric content **1120** stored in various databases.

This involves matching the tagged items in the temporary storage space with calendar items **1130** in the Calendar Database **1140**; email items **1115** in the Email Database **1114**; contact items **1117** in the Contact Database **1168**; task list items **1119** in the Task List Database **1118**; and news items **1121** in the News Database **1120**. After retrieving any relevant user-centric content, it is compiled together and returned **1122**.

User Persona

The system allows the user to create a number of different person as that aggregate profile information into sets that are

useful in different contexts. A user may create one persona when making purchases for his home. This persona may contain his home address and may indicate that this user is looking to find a good bargain when shopping. The same user may create a second persona that can be used when he is in a work context. This persona may store the user's work address and may indicate that the user prefers certain vendors or works for a certain company that has a discount program in place. When shopping for work-related items, the user may use this persona. A persona may also contain rules and restrictions. For instance, the work persona may restrict the user to making airline reservations with only one travel agent and utilizing booking rules set up by his employer.

FIG. **12** describes the relationship between a user, his multiple person as and his multiple profiles. At the User Level is the User Profile **1200**. This profile describes the user and his account information. There is one unique record in the database for each user who has an account. Attached to each user are multiple Personas **1220, 1230** & **1240**. These Personas are used to group multiple profiles into useful contexts. For instance, consider a user who lives in San Francisco and works in Palo Alto, but has a mountain cabin in Lake Tahoe. He has three different contexts in which he might be accessing his site. One context is work-related. The other two are home-life related, but in different locations. The user can create a Persona for Work **1220**, a Persona for Home **1230**, and a Persona for his cabin home **1240**. Each Persona references a different General Profile **1250, 1260** and **1270** which contains the address for that location. Hence, there are three General Profiles. Each Persona also references one of two Travel Profiles. The user maintains a Work Travel Profile **1280** that contains all of the business rules related to booking tickets and making reservations. This Profile may specify, for instance, that this person only travels in Business or First Class and his preferred airline is United Airlines. The Work Persona references this Work Travel Profile. The user may also maintain a Home Travel Profile **1290** that specifies that he prefers to travel in coach and wants to find non-refundable fairs, since they are generally cheaper. Both the Persona for Home and the Persona for the cabin home point to the Home Travel Profile.

FIG. **13** describes the data model that supports the Persona concept. The user table **1310** contains a record for each user who has an account in the system. This table contains a username and a password **1320** as well as a unique identifier. Each user can have multiple Personas **1330**, which act as containers for more specialized structures called Profiles **1340**. Profiles contain the detailed personal information in Profile Field **1350** records. Attached to each Profile are sets of Profile Restriction **1360** records. These each contain a Name **1370** and a Rule **1380**, which define the restriction. The Rule is in the form of a pattern like (if x then y), which allows the Rule to be restricted to certain uses. An example Profile Restriction would be the rule that dictates that the user cannot book a flight on a certain airline contained in the list. This Profile Restriction could be contained in the "Travel" Profile of the "Work" Persona set up by the user's employer, for instance. Each Profile Field also contains a set of Permissions **1390** that are contained in that record. These permissions dictate who has what access rights to that particular Profile Field's information.

### Intention-Centric Interface

Satisfying Customer Intentions, such as Planning for Retirement or Relocating requires a specialized interface. Customer Intentions require extensive planning and coordination across many areas, ranging from financial security, housing and transportation to healthcare, personal and professional development, and entertainment, among others. Satisfying Intentions requires a network of complementary businesses, working across industries, to help meet consumers' needs.

An Intention-Centric Interface is a user interface designed to help the user manage personal Intentions. At any given point, the interface content is customized to show only content that relates to that particular Intention. The Intention-Centric Interface allows the user to manage the process of satisfying that particular Intention. This involves a series of discrete steps and a set of content areas the user can access. At any point, the user can also switch the interface to manage a different Intention, and this act will change the content of the interface to include only that content which is relevant to the satisfaction of the newly selected Intention.

FIG. **14** provides a detailed description of the data model needed to support an Intention-Centric Interface. Each User Persona **1410** (see FIG. **13** for a more detailed description of the Persona data model.) has any number of active User Intentions **1420**. Each active User Intention is given a Nickname **1430**, which is the display name the user sees on the screen. Each active User Intention also contains a number of Data Fields **1440**, which contain any user data collected throughout the interaction with the user. For instance, if the user had filled out a form on the screen and one of the fields was Social Security Number, the corresponding Data Field would contain Name="SSN" **1450**, Value="999-99-9999" **1460**. Each User Intention also keeps track of Intention Step **1470** completion status. The Completion **1480** field indicates whether the user has completed the step. Every User Intention is a user-specific version of a Generic Intention **1490**, which is the default model for that Intention for all users. The Generic Intention is customized through Custom Rules **1411** and **1412** that are attached to the sub-steps in the Intention. These Custom Rules are patterns describing how the system will customize the Intention for each individual user using the individual user's profile information.

### Statistical Agent

An agent keeps track of key statistics for each user. These statistics are used in a manner similar to the Tamagochi virtual reality pet toy to encourage certain behaviors from the user. The statistics that are recorded are frequency of login, frequency of rating of content such as news articles, and activity of agents, measured by the number of tasks which it performs in a certain period. This information is used by the system to emotionally appeal to the user to encourage certain behaviors.

FIG. **15** describes the process for generating the page that displays the agent's current statistics. When the user requests the agent statistics page **1510** with the client browser, the server retrieves the users' statistics **1520** from the users' profile database **1530**. The server then performs the mathematical calculations necessary to create a normalized set of statistics **1540**. The server then retrieves the formulas **1550** from the content database **1560** that will be used to calculate the user-centric statistics. Graphs are then generated **1570** using the generic formulas and that user's statistics. These graphs are inserted into a template to create the statistics page **1580**. This page is then returned to the user **1590**.

### Personalized Product Report Service

The system provide Consumer Report-like service that is customized for each user based on a user profile. The system

records and provides ratings from users about product quality and desirability on a number of dimensions. The difference between this system and traditional product quality measurement services is that the ratings that come back to the users are personalized. This service works by finding the people who have the closest match to the user's profile and have previously rated the product being asked for. Using this algorithm will help to ensure that the product reports sent back to the user only contain statistics from people who are similar to that user.

FIG. **16** describes the algorithm for determining the personalized product ratings for a user. When the user requests a product report **1610** for product X, the algorithm retrieves the profiles **1620** from the profile database **1630** (which includes product ratings) of those users who have previously rated that product. Then the system retrieves the default thresholds **1640** for the profile matching algorithm from the content database **1650**. It then maps all of the short list of users along several dimensions specified in the profile matching algorithm **1660**. The top n (specified previously as a threshold variable) nearest neighbors are then determined and a test is performed to decide if they are within distance y (also specified previously as a threshold variable) of the user's profile in the set **1670** using the results from the profile matching algorithm. If they are not within the threshold, then the threshold variables are relaxed **1680**, and the test is run again. This processing is repeated until the test returns true. The product ratings from the smaller set of n nearest neighbors are then used to determine a number of product statistics **1690** along several dimensions. Those statistics are inserted into a product report template **1695** and returned to the user **1697** as a product report.

Personal Profile and Services Ubiquity

This system provides one central storage place for a person's profile. This storage place is a server available through the public Internet, accessible by any device that is connected to the Internet and has appropriate access. Because of the ubiquitous accesibility of the profile, numerous access devices can be used to customize services for the user based on his profile. For example, a merchant's web site can use th is profile to provide personalized content to the user. A Personal Digital Assistant (PDA) with Internet access can synchronize the person's calendar, email, contact list, task list and notes on the PDA with the version stored in the Internet site. This enables the person to only have to maintain one version of this data in order to have it available whenever it is needed and in whatever formats it is needed.

FIG. **17** presents the detailed logic associated with the many different methods for accessing this centrally stored profile. The profile database **1710** is the central storage place for the users' profile information. The profile gateway server **1720** receives all requests for profile information, whether from the user himself or merchants trying to provide a service to the user. The profile gateway server is responsible for ensuring that information is only given out when the profile owner specifically grants permission. Any device that can access the public Internet **1730** over TCP/IP (a standard network communications protocol) is able to request information from the profile database via intelligent HTTP requests. Consumers will be able to gain access to services from devices such as their televisions **1740**, mobile phones, Smart Cards, gas meters, water meters, kitchen appliances, security systems, desktop computers, laptops, pocket organizers, PDAs, and their vehicles, among others. Likewise, merchants **1750** will be able to access those profiles (given permission from the consumer who owns each profile), and will be able to offer customized, personalized services to consumers because of this.

One possible use of the ubiquitous profile is for a hotel chain. A consumer can carry a Smart Card that holds a digital certificate uniquely identifying him. This Smart Card's digital certificate has been issued by the system and it recorded his profile information into the profile database. The consumer brings this card into a hotel chain and checks in. The hotel employee swipes the Smart Card and the consumer enters his Pin number, unlocking the digital certificate. The certificate is sent to the profile gateway server (using a secure transmission protocol) and is authenticated. The hotel is then given access to a certain part of the consumer's profile that he has previously specified. The hotel can then retrieve all of the consumer's billing information as well as preferences for hotel room, etc. The hotel can also access the consumer's movie and dining preferences and offer customized menus for both of them. The hotel can offer to send an email to the consumer's spouse letting him/her know the person checked into the hotel and is safe. All transaction information can be uploaded to the consumer's profile after the hotel checks him in. This will allow partners of the hotel to utilize the information about the consumer that the hotel has gathered (again, given the consumer's permission).

Intention Value Network

In an Intention Value Network, the overall integrator system coordinates the delivery of products and services for a user. The integrator manages a network of approved suppliers providing products and services, both physical and virtual, to a user based on the user's preferences as reflected in the user's profile. The integrator manages the relationship between suppliers and consumers and coordinates the suppliers' fulfillment of consumers' intentions. It does this by providing the consumer with information about products and suppliers and offering objective advice, among other things.

FIG. **18** discloses the detailed interaction between a consumer and the integrator involving one supplier. The user accesses a Web Browser **1810** and requests product and pricing information from the integrator. The request is sent from the user's browser to the integrator's Web/Application Server **1820**. The user's preferences and personal information is obtained from an integrator's customer profile database **1830** and returned to the Web/Application server. The requested product information is extracted from the supplier's product database **1840** and customized for the particular customer. The Web/Application server updates the supplier's customer information database **1850** with the inquiry information about the customer. The product and pricing information is then formatted into a Web Page **1860** and returned to the customer's Web Browser.

Summary Agent

A suite of software agents running on the application and web servers are programmed to take care of repetitive or mundane tasks for the user. The agents work according to rules set up by the user and are only allowed to perform tasks explicitly defined by the user. The agents can take care of paying bills for the user, filtering content and emails, and providing a summary view of tasks and agent activity. The user interface for the agent can be modified to suit the particular user.

FIG. **19** discloses the logic in accordance with a preferred embodiment processing by an agent to generate a verbal summary for the user. When the user requests the summary

page **1900**, the server gets the user's agent preferences **1920**, such as agent type, rules and summary level from the user profile database **1930**. The server gets the content **1940**, such as emails, to do list items, news, and bills, from the content database **1950**. The agent parses all of this content, using the rules stored in the profile database, and summarizes the content **1960**. The content is formatted into a web page **1970** according to a template. The text for the agent's speech is generated **1980**, using the content from the content database **1990** and speech templates stored in the database. This speech text is inserted into the web page **1995** and the page is returned to the user **1997**.

### Trusted Third Party

The above scenario requires the web site to maintain a guarantee of privacy of information according to a published policy. This system is the consumer's Trusted Third Party, acting on his behalf in every case, erring on the side of privacy of information, rather than on the side of stimulation of commerce opportunities. The Trusted Third Party has a set of processes in place that guarantee certain complicity with the stated policy.

### "meCommerce"

This word extends the word "eCommerce" to mean "personalized electronic commerce." FIG. **20** illustrates a display login in accordance with a preferred embodiment. The display is implemented as a Microsoft Internet Explorer application with an agent **2000** that guides a user through the process of interacting with the system to customize and personalize various system components to gather information and interact with the user's personal requirements. A user enters a username at **2010** and a password at **2020** and selects a button **2040** to initiate the login procedure. As the logo **2030** suggests, the system transforms electronic commerce into a personalized, so called "me" commerce.

FIG. **21** illustrates a managing daily logistics display in accordance with a preferred embodiment. A user is greeted by an animated agent **2100** with a personalized message **2190**. The user can select from various activities based on requirements, including travel **2110**, household chores **2120**, finances **2130** and marketplace activities **2140**. Icons **2142** for routine tasks such as e-mail, calendaring and document preparation are also provided to facilitate rapid navigation from one activity to another. Direct links **2146** are also provided to allow transfer of news and other items of interest. Various profiles can be selected based on where the user is located. For example, work, home or vacation. The profiles can be added **2170** as a user requires a new profile for another location. Various items **2180** of personal information are collected from the user to support various endeavors. Moreover, permissions **2150** are set for items **2180** to assure information is timely and current.

FIG. **22** illustrates a user main display in accordance with a preferred embodiment. World **2200** and local news **2210** is provided based on a user's preference. The user has also selected real estate **2230** as an item to provide direct information on the main display. Also, a different agent **2220** is provided based on the user's preference.

FIG. **23** illustrates an agent interaction in accordance with a preferred embodiment. The agent **2310** is communicating information **2300** to a user indicating that the user's life insurance needs have changed and pointing the user to the chart that best summarizes the information for the user. Particular tips **2395** are provided to facilitate more detailed information based on current user statistics. A chart **2370** of the user's life insurance needs is also highlighted at the center of the display to assist the user in determining appropriate action. A button **2380** is provided to facilitate changing the policy and a set of buttons **2390** are provided to assist a user in selecting various views of the user's insurance requirements.

### Event Backgrounder

An Event Backgrounder is a short description of an upcoming event that is sent to the user just before an event. The Event Backgrounder is constantly updated with the latest information related to this event. Pertinent information such as itinerary and logistics are included, and other useful information, such as people the user knows who might be in the same location, are also included. The purpose of the Event Backgrounder is to provide the most up-to-date information about an event, drawing from a number of resources, such as public web sites and the user's calendar and contact lists, to allow the user to react optimally in a given situation.

### Vicinity Friend Finder

This software looks for opportunities to tell the user when a friend, family member or acquaintance is or is going to be in the same vicinity as the user. This software scans the user's calendar for upcoming events. It then uses a geographic map to compare those calendar events with the calendar events of people who are listed in his contact list. It then informs the user of any matches, thus telling the user that someone is scheduled to be near him at a particular time.

### Information Overload

The term information overload is now relatively understood in both its definition as well as its implications and consequences. People have a finite amount of attention that is available at any one time, but there is more and more vying for that attention every day. In short, too much information and too little time are the primary factors complicating the lives of most knowledge workers today.

The first attempts to dynamically deal with information overload were primarily focused on the intelligent filtering of information such that the quantity of information would be lessened. Rather than simply removing random bits of information, however, most of these approaches tried to be intelligent about what information was ultimately presented to the user. This was accomplished by evaluating each document based on the user's interests and discarding the less relevant ones. It follows, therefore, that the quality was also increased.

Filtering the information is only a first step in dealing with information is this new age. Arguably, just as important as the quality of the document is having ready access to it. Once you have entered a meeting, a document containing critical information about the meeting subject delivered to your office is of little value. As the speed of business continues to increase fueled by the technologies of interconnectedness, the ability to receive quality information wherever and whenever you are becomes critical. This new approach is called intelligent information delivery and is heralding in a new information age.

A preferred embodiment demonstrates the intelligent information delivery theory described above in an attempt to not only reduce information overload, but to deliver high quality information where and when users'require it. In other words, the system delivers right information to the right person at the right time and the right place.

Active Knowledge Management System Description

FIG. **24** is a block diagram of an active knowledge management system in accordance with a preferred embodiment. The system consists of the following parts: back-end **2400** connection to one or more servers, personal mobile wireless clients (Awareness Machine)**2430**, **2436**, public clients (Magic Wall) **2410**, **2420**, web clients **2446**, **2448**, e-mail clients **2450**, **2460**.

Back-end Server (2400) Processes

FIG. **25** is a block diagram of a back end server in accordance with a preferred embodiment. The back-end (**2400** of FIG. **24**) is a computer system that has the following software active: Intelligent Agents Coordinator (Munin) **2580**, Information Prioritization Subsystem **2530**, a set of continuously and periodically running information gathering and processing Intelligent Agents **2500**, **2502** and **2504**, User Profiles Database **2542** and supporting software, Information Channels Database **2542** and supporting software, communications software **2550**, information transformation software **2560**, and auxiliary software.

The Awareness Machine (2446 & 2448 of FIG. 24)

The Awareness Machine is a combination of hardware device and software application. The hardware consists of handheld personal computer and wireless communications device. The Awareness Machine reflects a constantly updated state-of-the-owner's-world by continually receiving a wireless trickle of information. This information, mined and processed by a suite of intelligent agents, consists of mail messages, news that meets each user's preferences, schedule updates, background information on upcoming meetings and events, as well as weather and traffic.

The Intelligent Agent Coordinator **2580** of FIG. **25** is also the user's "interface" to the system, in that whenever the user interacts with the system, regardless of the GUI or other end-user interface, they are ultimately dealing with (asking questions of or sending commands to) the Intelligent Agent Coordinator. The Intelligent Agent Coordinator has four primary responsibilities: 1) monitoring user activities, 2) handling, information requests, 3) maintaining each user's profile, and 4) routine, information to and from users and to and from the other respective agents.

Monitoring User Activities

Anytime a user triggers a sensor the Intelligent Agent Coordinator receives an "environmental cue." These cues not only enable the Intelligent Agent Coordinator to gain an understanding where users'are for information delivery purposes, but also to learn the standard patterns (arrival time, departure time, etc.) of each persons' life. These patterns are constantly being updated and refined in an attempt to increase the system's intelligence when delivering information. For instance, today it is not uncommon for a person to have several email accounts (work-based, home-based, mobile-based, etc.) as well as several different computers involved in the retrieval process for all of these accounts. Thus, for the Intelligent Agent Coordinator to be successful in delivering information to the correct location it must take into account all of these accounts and the times that the user is likely to be accessing them in order to maximize the probability that the user will see the information. This will be discussed further in another section.

Handling Information Requests

The Intelligent Agent Coordinator handles information requests from other agents in order to personalize information intended for each user and to more accurately reflect each user's interests in the information they are given. These requests will commonly be related to the user's profile. For instance, if an agent was preparing a traffic report for a user it may request the traffic region (search string) of that user from the Intelligent Agent Coordinator. All access to the user's profile data is accessed in this method.

Maintaining User Profiles

User profiles contain extensive information about the users. This information is a blend of user-specified data and information that the Intelligent Agent Coordinator has learned and extrapolated from each user's information and activities. In order to protect the data contained in the profiles, the Intelligent Agent Coordinator must handle all user information requests. The Intelligent Agent Coordinator is constantly modifying and updating these profiles by watching the user's activities and attempting to learn the patterns of their lives in order to assist in the more routine, mundane tasks. The Intelligent Agent Coordinator also employs other agents to glean meaning from each user's daily activities. These agents mine this data trying to discover indications of current interests, long-term interests, as well as time delivery preferences for each type of information. Another important aspect of the Intelligent Agent Coordinator's observations is that it also tries to determine where each user is physically located throughout the day for routing purposes.

Information Routing

Most people are mobile throughout their day. The Intelligent Agent Coordinator tries to be sensitive to this fact by attempting to determine, both by observation (unsupervised learning) and from cues from the environment, where users are or are likely to be located. This is certainly important for determining where to send the user's information, but also for determining in which format to send the information. For instance, if a user were at her desk and using the web client, the Intelligent Agent Coordinator would be receiving indications of activity from her PC and would know to send any necessary information there. In addition, because desktop PCs are generally quite powerful, a full-featured, graphically intense version could be sent. However, consider an alternative situation: the Intelligent Agent Coordinator has received an indication (via the keycard reader next to the exit) that you have just left the building. Minutes later the Intelligent Agent Coordinator also receives notification that you have received an urgent message. The Intelligent Agent Coordinator, knowing that you have left the building and having not received any other indications, assumes that you are reachable via your handheld device (for which it also knows the capabilities) and sends the text of the urgent message there, rather than a more graphically-oriented version.

Inherent Innovations

The Active Knowledge Management system represents some of the most advanced thinking in the world of knowledge management and human computer interaction. Some of the primary innovations include the following:

- The Intelligent Agent Coordinator as illustrated above.
- The development, demonstration, and realization of the theory of Intelligent Information Delivery
- Support for several channels of information delivery, all of which utilize a common back-end. For instance, if a user is in front of a Magic Wall the information will be presented in a multimedia-rich form. If the system determines that the user is mobile, the information will be sent by to their Awareness Machine in standard text. It facilitates delivery of information whenever and wherever a user requires the information.
- Personalization of information based not only on a static user profile, but also by taking into account history of the user interactions and current real-time situation including "who, where, and when" awareness.

Utilization of fast and scalable Information Prioritization Subsystem that takes into account Intelligent Agents Coordinator opinion, user preferences, and history of user interactions. It takes the load of mundane decisions off the Intelligent Agents part therefore allowing the agents to be much more sophisticated and precise without compromising the system scalability.

Speech recognition and speech synthesis in combination with intelligent agent animated representation and tactile input provides for efficient, intuitive, and emotionally rewarding interaction with the system.

## Client Reporting Subsystem Model

### Context

The Reporting subsystem is used by other subsystems on the client to report (read: make a matter of record) various data. The subsystem makes no assumptions about the type of data it handles—the data could be fault reports (as part of an architectural service) or lead management information (as part of an application data service). The Reporting subsystem is in this sense part of the infrastructure, it is an underlying set of services available everywhere on the client. The Reporting subsystem uses the Communications subsystem to store and forward data.

### Architecture Overview

The Reporting subsystem offers services to every client subsystem. It comprises a mechanism for messaging within the client application and between the client and the server.The Reporting mechanism uses the Communications subsystem to store and forward data. The Exceptions use this Reporting mechanism for reporting information about errors only.

### Role

The Reporting subsystem provides a set of infrastructural services which allow architectural components to report information. The subsystem makes no stipulation about the information reported, although it does contain components that map to certain types of reported information, such as system faults or customer interaction information. Part of the subsystem interface is presented as a set of Exceptions, which allow the automatic reporting of error conditions encountered during processing.

The subsystem accepts data and forwards it in an appropriate format to the Communications subsystem. It is captures and reports Exceptions generated during processing that are the result of error conditions. It is able to deal with any type of report that needs to be made, from error logging to sales leads. It is flexible enough to record new types of information as required. It is also flexible enough to be able to add new types of report as required. In addition, it is able to deal with the non-availability of certain information during the logging process.

### Responsibilities

The Client Reporting Subsystem is responsible for providing a set of Exceptions for use by all parts of the application. The Subsystem is also responsible for logging fault reports, user interaction reports, application heartbeat reports, message receipts, referrals or leads, and Management Information System entries.

### Exclusions

The subsystem is not responsible for gathering information from interface interactions or elsewhere; neither is it responsible for deciding what of a set of data needs to be reported. Reporting does not include the printing of reports.

| Component Specifications | |
|---|---|
| Client Exception Reporting Component | This is a set of Exception classes which, using the Client Reporting Component reporting services, stores and sends fault information |
| Client Reporting Component | This is the mechanism by which information for all reports are collected, formatted, and submitted to the Client Communications Subsystem. |

### Creation, Existence, and Management

The key element of the subsystem is a static class which manages the creation of report objects—a report factory. This class is instantiated by the Communications subsystem (which manages client configuration) and is always available. It is a severe error if it is not. Reports are generated on an ad hoc basis as needed.

### Sizing and Capacity

Whatever the requirements of the client architecture, the 'throughput' of the Client Reporting subsystem (understood as the number of reports, of every type, that are requested in a given time) will not place significant strain on system resources. Most of the capacity requirements for reports are absorbed by the Communications subsystem, which must arrange for the storage and transmission of those reports. Nevertheless, the subsystem must be able to deal with whatever throughput is demanded by the architecture, and the design takes into consideration the estimated workload generated by each part of the architecture.

### Performance

Performance is not critical for the Client Reporting subsystem. Reported data, with a few exceptions, is stored before transmission, and so a delay before data is sent is anticipated. Certain types of severe or critical faults need to be reported at once, but the low bandwidth required for these transmissions will not present performance problems.

### Design Guidelines

The reporting needs of the architecture fluctuate, although a core set of capabilities (fault reporting, lead management, interaction reporting) always remain requirements. The subsystem is flexible enough not only to extend or reduce its capabilities, but also to adjust the level of detail and the nature of data it records for each capability. Implementation of the system follows the project Java coding standards.

### Logical Components

Exceptions within the Client Exception Reporting component call the Client Reporting component to create fault reports.

| Component Descriptions | |
|---|---|
| Client Exception Component | This is a hierarchy of Exception classes structured to assist in the handling and passing of Exceptions. These Exceptions will accept information about the Exception event they represent. With this information and whatever else the class knows about it's own event, the component will use the Client Reporting Component to create fault reports. |
| Client Reporting Component | A static report factory will accept requests from other components in the form of a signalled event. Based on this event, the factory will manufacture a report of a certain type. The report will then be populated with information supplied by the calling component or reporter. |

Submitting a Report

Component A signals the Client Reporting subsystem to indicate that a reportable event has occurred. The Client Reporting subsystem then requests information about the event and creates a report. Finally, the Client Reporting subsystem signals to the Client Communication subsystem that the report is ready to be sent.

Throwing an Exception

First, Component A signals that an exceptional event has occurred by instantiating an appropriate Exception. Second, Component A passes reportable event relevant information to the Exception. Third, Component A requests for the Exception to be thrown. Fourth, the Client Exception Reporting component submits a report to be sent based on the information available using the Client Reporting component as outlined above. Finally, the Client Exception Reporting component throws it's Exception.

Local Content Subsystem Model

Context

The Local Content subsystem provides all content required by the application. This includes both static and dynamic content. It also provides business services required by the application. It operates on a "storage and retrieval" basis, storing the data obtained from the user and the business, and providing mechanisms to retrieve that data. The Local Content subsystem is used by the ISF subsystem to provide content. It uses the Communication subsystem to receive business data.

Architecture Overview

Objects in the Local Content subsystem are created by the Initialization subsystem of the Application Architecture. Services provided by the Local Content subsystem are also accessed through the Application Architecture, via the Initialization and ISF subsystems.

The example components of the Local Content subsystem reflect different types of business knowledge and processes required by the application. User Data and Business Data involve data collected respectively from the user and the business. The Calculation component performs complex calculations, and the Product component represents the products used by the business. The Content Providers component defines static media content.

Role

The Local Content subsystem provides static and dynamic content to the application. It also provides all business-specific services required by the application.

Responsibilities

The Local Content subsystem provides static and dynamic media to the application. The Local Content subsystem also stores store user entered details, business data, and performs business calculations.

Exclusions

All application-specific behaviour is provided through the Application Personality. This behaviour is defined by Hamlet scripts, which also define navigation between scripts. The scripts are divided into metaphors, each of which is an embodiment of a style of interaction.

Access to media is provided through the Content Providers component, which belongs to this subsystem. However, the objects of this component are created automatically by the System Initialiser component from a contents file defined in the Application Personality.

Creation, Existence, and Management

The Business Data component is created and initialized at the time of System Initialization. It exist for the life of the system.

The User Data component is available for the entire time a customer is using the system. When a customer session ends, references to all objects in the User Data component are released so that the objects can be garbage collected.

Objects in the Local Content subsystem (with the exception of the Business Data Component) are created and initialized either internally or by the Initialization subsystem. References to these objects are managed by the ISF subsystem. Business Data objects are created, initialized and managed by the Communications subsystem.

Logical Components

The ISF does not actually know about the Local Content subsystem. The Local Content subsystem implements a set of interfaces defined by the ISF. The Initialization subsystem uses the scripts defined in the Application Personality to define which objects (implementing those interfaces) need to be used to retrieve content. The ISF uses the Initialization component to create those objects, then manages them.

Developers of the Application Personality can easily view their scripts as directly managing the local content objects. This allows the local content objects to be developed without knowledge of the Application Architecture layer.

Component Descriptions
Interface Support Framework Subsystem Model

| | |
|---|---|
| User Data | Stores and retrieves data entered by the user, and initiates calculations on stored data. |
| Business Data | Stores and retrieves data provided by the business. |
| Calculation | Performs complex calculations. |
| Product | Provides access to information associated with particular marketing products. |
| Content Providers | Provides access to static media. This is an implementation of an interface and is not documented as a separate component. |

Context

The Interface Support Framework subsystem is part of the Application Architecture Layer. It provides a rich interactive environment which exploits the full potential of a dynamic, multi-media interface. The ISF is built around a theatrical metaphor where every object is expected to exert dynamic behavior.

Objects within the ISF are initialized by the Application Initialization subsystem within the Application Architecture Layer and utilize the services of the Content Players, Printing and Reporting subsystems in the Technical Architecture Layer.

Architecture Overview

Objects within the ISF subsystem are initialized by the Application Initialization Subsystem. Reporting and Transaction Interface Services are used to log ISF data for the Technical Architecture layer to report or print. The Content Player subsystem within the Technical Architecture is used by the ISF to present media to the user.

The ISF subsystem is built upon a layered architecture which follows the Model-View-Controller pattern. The Factual component contains the object model of the business and definitions of business media content. The Visual component displays and manipulates media to provide a view of the business model to the user. The Behavioral component controls all interactions between the Visual and Factual components.

Role

The ISF subsystem provides the services for the application to present multimedia content in a controlled way. It also provides the capability to react to user input and affect changes to the scene.

Responsibilities

The ISF subsystem displays each scene of the application, and modifies the content of a scene while it is displayed. In addition, the ISF subsystem enables navigation between scenes, reacts to user interaction, retrieves business content, performs business functions and calculations, and provides common user interface constructs. Other responsibilities of the ISF include initiating print jobs and video conference sessions, reporting on user entry into a scene, duration of a session, user interaction with a role, user navigation, and reporting on errors occurring within the ISF. Finally, the ISF manages user sessions, and responds to system level events such as system start up and shutdown or screen paint requests.

Design Guidelines

The ISF subsystem provides the services for the application to present multimedia content. It is architecturally layered into three distinct components which parallel the Model-View-Controller paradigm. This separates the core business objects and their data (the Model), from the visual representation of this information (the View), from the logic to control and react to changes in the Model or the View (the Controller). The architecture provides boundaries between the graphical style of the system (Stage, Roles and Scenes), the operational code (Actors, Scene Director and Stage Manager), and the underlying Content Providers (Business Objects). These sections are the Visual, the Behavioral, and the Factual components.

The Factual Layer is not aware of the Visual layer. This allows the visual metaphor to change, without disrupting the underlying business domain model. The Behavioral level mediates between the Factual and visual layers and should avoid very complex interactions with either layer. Where possible, anonymous communications via a Publish/Subscribe pattern is used to avoid further interdependencies between the layers.

The Stage is identified as the display context. It is able to communicate only with the Locations it controls. It is hidden behind the StageManager, where all visual requests need to be managed.

The ISF is a layered system. All roles in a scene form a series of visual siblings. These roles can, in fact, contain and encapsulate other roles. This allows, through recursion, any number of distinct processing layers. Each child only communicates with its direct parent, surrendering control of communicating beyond to the parent. This containment relationship is possible in both the Visual and Behavioral layers.

To assist in navigation, Scene Thumbnails are maintained. The user may touch on a Scene Thumbnail to return to a previously visited Scene.

| Component Descriptions | |
|---|---|
| Visual component | user interaction and presentation of multimedia content |
| Behavioral component | application behavior and multimedia content retrieval |
| Factual component | provides multimedia content and business function services and calculations |

Process Control

This table describes the various key threads which execute within the ISF.

| Thread | Purpose |
|---|---|
| AWT | Sends windows messages (e.g., screen touches) to the ProcessController. This thread is created by the Java Virtual Machine. |
| Main | Initializes the application, then exits. This thread is created by the Java Virtual Machine. |
| Processing | Performs the actions initiated by the ProcessController. |
| Timer | Generates and actions time based system events such as session timeouts. |
| Video Status | Receives notification of video finish events and dispatches them to the ProcessController. |
| Audio Status | Receives notification of audio finish events and dispatches them to the ProcessController. |

User Touches Location on Stage

Description

End users will touch the visible window of the application, the Stage. This will initiate a response from the application.

Actors: End User Components Involved: Visual

Key Objects Involved: Stage

Stage processes User Touch

Description

The application window will determine which location is affected by the touched area, and will notify the corresponding Role of a touch. The stage will also control the visual cue displayed on the window.

Actors: End user Components Involved: Visual

Key Objects Involved: Timer, User Interaction Reporter, Location, Media Player

Role Accepts User Touch Event

Description

Each role is notified of a user touch. This will force it to request a media change in its corresponding media, and, once accomplished, to notify its actor of the user interaction.

Actors: Stage

Components Involved: Visual, Behavioral

Key Objects Involved: Actor, Role

Actor activates Event Casting

Description

The actor will cycle through all of its registered Casting Lists and activate all castings which are interested in the specific event. Castings behave polymorphically, and therefore the behaviour of how to respond is actually held in the Casting, not the actor.

Actors: Role, Execute Casting

Components Involved: Behavioral

Key Objects Involved: Actor, Casting, Stage Manager, Scene Director

Stage Manager Performs Scene Transition

Description

The Stage Manager will replace the currently active scene with a new scene, based on the information in the Navigation Casting. It is important to control how the change occurs, to preserve the visual illusion of the Kiosk World. This scenario is also invoked when starting, or restarting, the application. In this case, there is no current Scene, but the application is told to transition to the first Scene of the application.

Actor: Navigation Casting, System Initializer

Components Involved: Behavioral, Visual

Key Objects Involved: Stage, Scene Director, Session Manager

Scene Director performs Cast Change

Description

The Scene Director coordinates the activation of all timings to ensure that any messages to the Stage Manager are grouped together and engaged at an appropriate time. This will ensure that all changes to the Roles visible on the scene will occur at the same time.

Actors: Content Casting, Slide Casting

Components Involved: Visual, Behavioral

Key Objects Involved: Scene Director, Stage Manager

Scene Director Performs Cast Change with Slide Effect

Description

The Scene Director must onstage all non-sliding Roles and then display the content of the Slide Casting along a straight line until its final destination.

Actors: Slide Casting

Components Involved: Visual, Behavioral

Business Object Invokes Business Function

Description

The application may request business information from the Business Domain Model (e.g., return a Repayment Amount). This interface also supports putting values into the business objects (such as store a Loan Term Amount). Each business object is provided a generic interface to invoke behaviors. The desired behaviour is specified in the Business Function Casting, and it capable of returning information to the Actor associated with the Business Object.

Actors: Business Function Casting

Components Involved: Factual

Stage Manager Times Outfrom Inactivity

Description

The application must support a time out facility, in the event that the user walks away from the application prior to returning to the Attractor Screen. This will protect the privacy of details entered by users.

Actors: Wait Timer

Components Involved: Behavioral

Session Manager Resets Application

Description

During the execution of the application, it may be necessary to reset the Stage to the first scene, and clear out the session. This may be triggered by system inactivity, or by direct user request dispatched through a business object.

Actors: Stage Manager's Wait Timer, Business Object

Components Involved: Behavioral

Reset Session Information

Description

Each session stores information gathered about the user. At the end of a session, or by user request, it is possible to erase all entered data. This supports privacy.

Actors: Session Manager

Components Involved: Behavioral

Responsive Media Displays Media

Description:

The Visual component of the ISF is responsible for displaying all media (image, audio, video, text) to the Stage. It actually interfaces with the underlying media subsystem in the Client Technical Architecture. Each player is obtained through the Gatekeeper, and supports all code required to present the media to the user.

Actors: Role

Components Involved: Visual, Behavioral, Factual, Content Players

Key Objects Involved: Responsive Media, Media Player, Stage

Application Requests Hard Copy Printout

Description

The client application may request a print out of static information (check list), or dynamic information (product explanation including current interest rates and other dynamic components of the product, product simulation and/or line graph). This is initiated by the end user, through a Print Casting.

Actors: Print Casting

Components Involved: Behavioral, Printing

Key Objects Involved: Business Object

Reporting Interface Subsystem Model

Context

The Reporting Interface Subsystem collects information logged by the Application Architecture Layer and sends it to the Client Reporting Subsystem in the Technical Architecture Layer.

Architecture Overview

The Reporting Interface information is logged by components within the Application Architecture Layer and sent to the Client Reporting Subsystem in the Technical Architecture Layer.

Role

The Reporting Interface subsystem provides services to log user-interaction with the kiosk and report on software and hardware faults which occur within the Application Architecture Layer.

Responsibilities

The Reporting Interface subsystem is responsible for gathering and logging user interaction with the kiosk by capturing what a user is doing with the system, e.g., which scenes they are visiting, which visual elements they are interacting with. The Reporting Interface subsystem also Gathers and logs information related to business products which the customer is interested in, the output of business functions which the customer has invoked and business data which the customer has input. Finally, the Reporting Interface subsystem captures information relating to the software and hardware performance of the kiosk. This information can then be used for error handling and fault management analysis.

Exclusions

The Reporting Interface subsystem does not include services to gather and log customer related information such as the a customer name and telephone numbers.

Systems Management Subsystem Model

Context

Systems Management involves the definition of a combination of automated and manual procedures. Automation is achieved primarily through the use of Systems Management Server (SMS). SMS is a tool within the Microsoft backoffice suite of tools which can centrally manage system software and hardware in a distributed environment.

Systems Management Subsystem Architecture

The Systems Management Subsystem architecture consists of two components, the Systems Management Server (SMS), and Fault Monitoring.

Systems Management Server (SMS)

Systems Management Server (SMS) is a Microsoft tool that can be used to distribute software/content, take software audits, perform fault diagnosis and take remote control. SMS is supplemented by a component developed for the architecture, namely the File Transfer Utility.

Fault Monitoring

The Kiosk is monitored real-time through a Heartbeat message system. Heartbeat pulses are sent from the Kiosk at a configurable rate (say one every minute) and are monitored at a console running the Kiosk Monitoring Application. If the status of a Kiosk changes to indicate a fault, monitoring application will initiate the appropriate action. Some errors

will be handled through the existing Operations Center. The routing of these errors is covered in the Application Services Subsystem.

Role

SMS is used for medium sized software distribution (both code and content) and for fault diagnosis of the remote kiosks. There are a large number of features that make SMS a flexible and useful support facility. Fault monitoring will provide the means to view the real-time status of each kiosk and associated peripherals. When a problem occurs at a particular kiosk (such as running out of paper), the kiosk will be brought to the attention of a operations representative. It will be possible to observe the kiosk status to verify the resolution of the problem.

Responsibilities

The SMS is responsible for software distribution, hardware fault management and diagnosis, and client remote re-boots. The SMS also provides a user interface to selectively view the status of all kiosks in the network.

Creation, Existence, and Management

The SMS resides on a dedicated server in accordance with a preferred embodiment and is available at all times.

Performance

The elapsed time between a fault occurring on an MMT and subsequently being displayed to operations is dependent upon the frequency of the heartbeat, the ability for the application server to process the heartbeat and the frequency of refresh on the operations terminal. An example in accordance with a preferred embodiment is presented below.

| | Elapsed Time (minutes) |
|---|---|
| A fault occurs immediately after a heartbeat message is sent. | 1:00 |
| The Heartbeat message is sent to the Application Server | 0:01 |
| The Application Server receives and processes the heartbeat message | 0:01 |
| The operations Fault Monitoring Application has just refreshed and is only refreshing once every 2 minutes. | 2:00 |
| The Fault Monitoring Application refreshes it's kiosk status main window | 0:05 |
| The Fault Monitoring Application refreshes it's kiosk status view (This time is for only one view window open. If there are more than one view windows open this time should be multiplied by the number of open view windows) | 0:05 |
| Total | 3:11 |

If a heartbeat message is not received from a kiosk within five minutes, (configurable) the Fault monitoring will set the kiosk status to Unknown. This time lag is required to avoid erroneously reporting MMT machines as unknown when the real problem lies in a slightly slower than normal processing of a heartbeat message.

Logical Components

An SMS site comprises of two components—a Primary Site and Clients. A primary site is the top most level in the SMS hierarchy. It contains its own SQL database to store system and inventory information for itself and other secondary sites underneath it. Clients (kiosks) are administered from the primary site. The client sends its hardware/software information to SMS server through the SMS Inventory service.

Fault Monitoring

After a configurable time interval the Client takes a status check of the Machine, Printer and the Application and sends it to the server in a heartbeat message. The server then places the status into a Kiosk Status Database that is monitored by operations staff for faults.

Server Communications Subsystem Model

Context

The Server Communications subsystem is part of the Server Technical Architecture. The Server Communications subsystem handles all communications between clients, the central server, and the mainframe host.

Architecture Overview

The Asynchronous Messaging component provides the asynchronous message based communication between the client and the server, using standard Internet mail protocols, SMTP and POP3.

The Business Process Access Module component provides the common point to invoke predefined business functionality such recording interaction information from the MMT. HTTP is the protocol used to communicate with HTTP servers on the World Wide Web. It is used in the MMT to distribute small updates of application components and content during the client configuration process on start-up. Access to mainframe database resident data is done by replicating the required database tables to corresponding server resident database tables. The reverse process is used to centrally store the data accumulated on the server to the mainframe database tables. Generic alerts from the server are transmitted to the mainframe through an interface to the mainframe's front end processor.

Role

The role of the Server Communications subsystem is to Provide the server with communications facilities between the MMT (or Internet) client and the network server and between the network server and the mainframe systems. In addition, the Server Communication subsystem isolates and provides access to organization specific functionality.

Responsibilities

The Server Communication subsystem complies with standard Internet protocols, to allow ease of porting to that delivery channel. In addition, The Server Communication subsystem provides reliable asynchronous communication between the client and the application server, and controlled and reliable access to organization specific functionality. Finally, the Server Communication subsystem provides a facility to deliver updates to the configuration of the client, such as application components and content, and provides access to the organization's legacy systems through predefined processes, such as database replication and generic alert reporting.

Exclusions

Server Communication is confined to invoking modules which conform to the ACT messaging architecture. If communication to another platform is required, this must be located within the external systems module using the organizations messaging or access methods.

Transaction Interface Subsystem Model

Architecture Overview

Transactions are initiated by components within the Application Architecture Layer and sent to the Client Reporting Subsystem in the Technical Architecture Layer. The Transaction services identified in this document are not implemented as separate components in their own right, but are implemented as extensions to existing application Architecture components.

Role

The Transaction Interface subsystem is responsible for providing an interface to the application for storing of contact information about the end-user. These include, but are not limited to information required to complete a loan, survey-based information on customer demographics, account balance inquiry, and funds transfer.

Customer Lead Transaction Execution

The customer lead transaction execution application facilitates the Interface Support Framework and enables the services of the Reporting Subsystem of the Technical Architecture to support the gathering and storing of information about the end-user.

Exclusions

The current usage of the Transaction Interface assumes that only asynchronous communications are available.

Server Application Services Subsystem Model

Architecture Overview

The Application Services Sub-system includes a set of definitions for building the MMT server application and architecture modules.

Role

The Server Application Services sub-system includes the set of services and definitions for accessing application and architecture functionality. This subsystem defines the structure and support services for building and executing applications and modules on the Application Server or Operations workstation platform. The functionality supported includes transaction processing t to/from the MMT: such as customer referral information, customer interaction information, MIS information, fault information, product rates and prices information, application configuration information, message receipt information, and heartbeat status information.

Responsibilities

The Server Application Services sub-system processes application business logic on the server independently from the underlying database management system. The application business logic includes customer referral information, customer interaction information, MIS information, fault information, product rates and prices information, application configuration information, message receipt information, and heartbeat status information.

The Server Application Services sub-system also provides a common service available to all server and client applications to log an error, decode a given code (for example, '1'=NSW, '2'=QLD etc.), and retrieve configuration information from the registry located on each machine.

Finally, the Server Applications Services sub-system invokes a Business Process (BP) for a given BP message.

Logical Components

The Server Application Services sub-system includes Common Servers, Data-Access Module, Business Process, and the Business Process Access Module. Definitions of each component are given in below.

Component Descriptions

Definitions

Common Services. Common services to support the development of application functionality include decoding codes tables, retrieving configuration information from the registry, message handling, and the support for logging and handling server application errors.

Data Access Module. A data access module (DAM) provides access to data within the application database. A DAM performs specific data access such as Insert, Delete. Update, Select, Select All across one or more tables. It is an MFC Extension DLL encapsulating a Recordset object which uses ODBC to access the underlying DBMS. The DAM definition outlines how these modules are used and coded.

Business Process. A business process (BP) is the application or architecture functionality that may be invoked by the Business Process Access Module architecture. A BP is identified by a message type. A BP accepts a request message defined by the BP and may provide a response message for synchronous messages. Database access is provided to the BP by DAMS. A BP is an MFC extension DLL with a defined entry point.

Business Process Access Module. This component is detailed in the communications sub-system. The Business Process Access Module (BPAM) is the architecture component that provides access to business processes. The BPAM invokes a BP for a given message type. The BPAM accesses the message address table to lookup BP module details. This component is detailed in the communications sub-system.

Wireless Electronic Valet in Accordance With A Preferred Embodiment

One embodiment of the present invention is an a Mobile Portal Platform including a Mobile Portal and an Electronic Valet. The Electronic Valet is a hand held wireless computer device executing Thin Client Software. Integrated into the Electronic Valet are various sensors, such as GPS, Biosensors, and Environ-sensors. In addition, recording equipment, such as a camera and auido recorder, is also integrated into the Electronic Valet. The Mobile Portal includes a Mobile Portal Server which is connected to various third party content and service providers through the Internet or a Mobile Portal Extranet.

FIG. **26** is a flow chart illustrating how the hardware and software of one embodiment of the present invention operates. An Electronic Valet **2602** receives input data from sensors, GPS, camera, microphones, and other user inputs **2600** integrated with the wireless hand held device. The Thin Client application executing on Electronic Valet **2602**, as discussed in detail below, allows the Electronic Valet **2602** to execute many different software applications without the need for a large amount of internal memory and storage capacity. The Electronic Valet **2602** forms a message based on the data received and the user input. The Electronic Valet **2602** then transmits the message via antennae **2604** to the Mobile Portal **2606**. The Mobile Portal **2606** parses the message received from the Electronic Valet **2602** and forms a new message based on the message received. The Mobile Portal **2606** then determines the appropriate third party service provider **2608** to transmit the new message to, based on the content of the message received from wireless hand held device **2602**, and then transmits the new message. The third party service provider then performs the appropriate service and transmits the result back to the Mobile Portal **2606**. The Mobile Portal then forms a message based on the data received from the third party service provider **2608** and transmits the message back to the Electronic Valet **2602**. The Electronic Valet **2602** then formats and displays the data received. The Electronic Valet **2602** utilizes a wireless modem such as a Ricochet SE Wireless Modem from Metricom.

Of course, wireless performance isn't nearly as reliable as a traditional dial-up phone connection. We were able to get strong connections in several San Francisco locations as long as we stayed near the windows. But inside CNET's all-brick headquarters, the Ricochet couldn't connect at all. When you do get online, performance of up to 28.8 kbps is available with graceful degradation to slower speeds. But even the slower speeds didn't disappoint. Compared to the alternative—connecting via a cellular modem—the Ricochet is much faster, more reliable, and less expensive to use. Naturally, the SE Wireless is battery powered. The modem has continuous battery life of up to 12 hours.

Thus, utilizing the wireless modem, a user may utilize the Mobile Portal **2606** via the Electronic Valet **2602**. Using appropriate key(s), the user may select a service to use in concert with appropriate data obtained from sensors, GPS, camera, microphones, and other user inputs **2600**. In certain circumstances, data may be automatically sent to select services based on the type and value of the data obtained by the Electronic Valet **2602**. For example, when an integrated bio-sensor obtains certain predefined data values, an appropriate emergency care provider would be automatically contacted. In addition, the data obtained from sensors, GPS, camera, microphones, and other user inputs **2600**, may also be combined before being sent to an appropriate service provider. For example, in the example above, GPS position data may be sent with the bio-sensor data to the emergency care provider. The emergency care provider would then know the patient's biological data and the location of the patient. Appropriate service could then be provided.

Mobile Portal Platform

The Mobile Portal Platform is a high-impact, server-based application in accordance with a preferred embodiment that is focused on the theme of delivering services and providing a personalized experience for each customer via a personal site located on a server. The services are intuitively organized around satisfying customer intentions—fundamental life needs or objectives that require extensive planning decisions, and coordination across several dimensions, such as financial planning, healthcare, personal and professional development, family life, and o ther concerns. Each member owns and maintains his own profile, enabling him to create and browse content in the system targeted specifically at him. From the time a demand for services is entered, intelligent agents are utilized to conduct research, execute transactions and provide advice. By using advanced profiling and filtering, the intelligent agents learn about the user, improving the services they deliver.

A preferred embodiment of a system utilizes a Windows CE PDA equipped with a GPS receiver. The embodiment is configured for a mall containing a plurality of stores. The system utilizes a GPS receiver to determine the user's location. One advantage of the system is that it enables the retrieval of data for nearby stores without relying on the presence of any special equipment at the mall itself. Although the accuracy of smaller, inexpensive receivers is limited to approximately 75–100 feet, this has thus far proven to be all that is necessary to identify accurately the immediately surrounding stores. The system uses generated data rather than actual store ads and prices. Well structured online catalogs are used. Other embodiments utilize agents that "learn to shop" at a given store using a relatively small amount of knowledge. Moreover, as retailers begin to use standard packages to create online catalogs, we can expect the number of differing formats to decrease, resulting in a tractable number of competing formats. As electronic commerce progresses, it is not unreasonable to expect standards to evolve governing how merchandise offerings are represented.

Goal Specification

Before leaving on a shopping trip, a shopper creates a shopping list of items by selecting from a preexisting set of approximately 85 product categories (e.g. men's casual pants, women's formal shoes, flowers, etc,). They also indicate the shopping venue they intend to visit from a list of malls.

Initial Store Selection

Upon arriving at the mall, begins by suggesting the closest store that sells at least one item of a type entered by the user during goal specification. Along with the store name a system in accordance with a preferred embodiment prepares a list of the specific items available and their prices. A map of the mall displays both the precise location of the store and the shopper's current location. The shopper queries the system to suggest a store at any time based on their current location.

Browsing

To address the need of many shoppers to visit malls or shop generally without a particular destination in mind. FIG. **27A** illustrates a display in accordance with a preferred embodiment of the invention. The display operates in a browse mode for use by shoppers as they stroll through the mall. In browse mode the system suggests items of interest for sale in the stores currently closest to the shopper. An item is considered to be of interest if it matches the categories entered in the goals screen. If there are no items of interest, the general type of merchandise sold at that store is displayed, rather than specific items. As the shopper strolls a map displays his or her precise current location in the mall. If an item displayed is selected by the shopper while browsing, the system alerts the shopper to the local retailer offering the same product for the lowest price, or announces the best local price. This search is restricted to the local mall, as that is the assumed radius the shopper is willing to travel.

Alternatives

It is worth emphasizing that the current inventive agent will support broader aspects of the shopping task, for example, it could operate as bi-directional channels. That is, not only can they provide information to the shopper, but, at the shopper's discretion, they may provide information to retailers as well. In this embodiment, the system indicates a shopper's goals and preferences to a retailer-based agent, who, in turn, responds with a customized offer that bundles service along with the product. Enabling the customization of offers is crucial to gaining the cooperation of retailers who are reluctant to compete solely on price and of value to customers who base their purchases on criteria other than price. While the preferred embodiment focuses on location-based filtering primarily in the context of the shopping task, the current invention provides the basis for "physical task support" agents that provide an information channel to people engaged in various tasks in the physical world.

The Predictive Value of Location

The present invention is a significant advance over non location based agents because a users physical location is often very predictive of his or hers current task. If we know someone is at a bowling alley or a post office we can reasonably infer their current activity. Knowledge of a user's current task largely determines the type of information they are likely to find useful. People are unlikely to concern themselves with postal rates while bowling, or optimal bowling ball weight while buying stamps. In addition, knowledge of the resources and obstacles present at a particular location suggest the range of possible and likely actions of someone at that location. This awareness of a user's possible and likely actions can be used to further constrain the type of information a user is likely to find useful. For example, knowledge of a restaurant's wine list could be used by a recommended system to constrain the wine advice it presents.

Knowledge of a shopper's precise location in a shopping mall is valuable because it enables the identification of the stores immediately surrounding the shopper. The offerings of the stores closest to the shopper represent the immediate choices available to the shopper. Given that shoppers place a premium on examining merchandise first hand and that

there is a cost associated with walking to other stores, the merchandise of the closest surrounding stores constitute the most likely immediate selections of the shopper. Consequently, among the most useful information provided at any given time is the availability of merchandise in the surrounding stores that matches their previously stated goals.

People tend to move to different locations while performing many of their our tasks. This suggests that their immediate surroundings do not completely capture the full range of options they may have. In fact one of the main reasons for leaving a location is to perform an action that is not possible at the current location.

Nevertheless, one does tend to address most tasks within relatively local areas. Thus while their immediate surroundings suggest the options they have available at a given point in time, a broader view of a location will often capture the options they are likely to consider over the course of a task. In the case of mall shopping, for example, the stores immediately surrounding the shopper represent the options available at that moment. Mall shoppers, however, are generally willing to travel to any store within the mall. Therefore the potential options over the entire shopping trip include all the stores in the mall. Accordingly, information is presented on offerings of interest only from the immediately surrounding stores because these are the immediately available options. When asked for alternatives, the system restricts itself to all the stores within the mall—the area within which the shopping task as a whole is likely to be performed. Being alerted that a store hundreds or thousands of miles away sells the same merchandise for a few dollars less than the cheapest local alternative is of little value in cases when shoppers require a first hand examination of the merchandise in question or are not willing to wait for shipping.

Physical vs. Online Shopping

In addition to the significant advantages over non-location based agents the present invention over comes disadvantages o online (or web) shopping. It is tempting to argue that online shopping will soon become the predominant mode of shopping, pending only greater penetration of home computers, the expansion of online offerings, and better online shopping tools. At first glance it would therefore appear to be a mistake to begin using location to support an activity that will become virtualized. Already we've seen the emergence of a number of software agents that support online shopping. For example, programs that allow users to identify the cheapest source for a music CD, given a title. Similar programs have been developed for buying books, such as BargainBot. These systems demonstrate the potential of electronic commerce web agents to create perfect markets for certain products. The success of these agents will encourage the development of similar web shopping agents for a greater variety of goods.

The Limitations of Online Shopping

Certainly online shopping will continue to grow and the trend towards more powerful online shopping agents will continue. Nevertheless, it also seems clear that no matter how sophisticated web-agents become, traditional physical shopping will continue to dominate the market for the foreseeable future. Several inherent difficulties of online shopping will ensure the continued reliance on physical shopping:

Non-fungible goods

Web-based shopping agents have typically enabled users to identify the cheapest price for fungible products such as books and music CDs. While this capacity to create "perfect markets" for such commodities is of great benefit to consumers, several difficulties exist that will complicate applying these approaches to arbitrary products.

Commodities are particularly well suited to shopping agents because it is easy to make comparisons between competing offers. Because commodities are fungible, one of the very few dimensions upon which they differ is price. Price therefore becomes the primary, if not sole, criterion upon which purchasing decisions are made.

As soon as we move beyond commodities, however, several other criteria become important. For example, how do we compare items such as sweaters, mattresses, or tables? In addition to price we care about the materials used, the color, how it fits and feels, and the workmanship. Similar problems apply to most other products.

Imprecise goal specification

A second, related difficulty lies in communicating our desires to an agent. Shopping agents are great if the user knows the precise commodity he or she wants. Then they can simply enter the product by name. Unfortunately, if they don't have a specific item in mind when they shop, then the problem of conveying what is wanted to an agent becomes more difficult. For example, how does the user tell an agent what kind of lamp they want for their living room?

Undeveloped Preferences

Interfaces that allow shoppers to include descriptive features like price ranges, color, options, brands, etc, can help address the above problem, but they are not enough. Much of the time shoppers either haven't formed preferences or can't articulate their desires until after they've started shopping and had a chance to examine various examples of the target products.

Shopping is Entertainment

People like to shop and do so without having a specific purchase in mind. One study found that 42% of consumers are "non-destination shoppers" that visit the mall primarily for leisure browsing and socializing.

Shopping is Sensory

Even if the user could effectively provide these details most would be unlikely to delegate a purchasing decision to such an agent. After all, many people are uncomfortable even trusting spouses to make appropriate purchases on their behalf. Most people want to see and touch first hand what they're considering before making a purchase decision. The few preferences they may provide an agent cannot replace this rich, first-hand experience. At best such preferences could be used to generate a candidate set for shoppers to consider.

Instant Gratification

Shopping is often a very emotional activity. People are pleased with their purchases and often can't wait to get home to try them out. The inherent delay between online purchases and their receipt is a significant issue to those who simply must take home their selections as soon as they see them.

In the end, consumers will continue to engage in physical shopping because of the limitations listed above. However, the fact that the task can't completely be delegated to software agents does not rule out a role for them. First, users find them useful for purchasing commodities when they know what they want. A second role, however, is to support the physical shopping task itself, throughout the time that a person is engaged in it. This, of course, is the approach taken in the SHOPPER'S EYE project.

Shopper's Eye

At first blush it may seem that the current invention is subject to some of the same limitations as purely web-based agents. After all, why should it be any easier to communicate

your goals to a PDA than it is to a web-based agent? Why would your preferences be any more developed for purchases supported by a PDA system than a web-based agent?

A key difference between purely web-based agents and the current "physical task support agents" (i.e. an agent that supports a user engaged in a task in a physical setting) is that web-based agents are completely responsible for conveying all information that will be considered by the user. On the other hand, "physical task support" agents in accordance with a preferred embodiment can augment the approaches of web-based agents by referring to aspects of a user's environment. For example, it is not terribly important to convey richly the feeling of a particular sweater if the sweater is in a store thirty feet away. It need only refer the shopper to the sweater. The shopper will gain a much better appreciation of the sweater by trying it on than through anything that can be conveyed by the system. When too many products match an imprecisely specified goal for a web-based agent, a more restrictive search must be made. However, many matches simply indicates there is a store that is likely to be of great interest to the shopper and therefore should be visited. Once inside, narrowing down the merchandise of interest in person will often be far easier than refining the goals on a web-based agent. Therefore physical task support agents can assist users to elaborate their preferences and identify specific goals by calling users' attention to aspects of their physical environment as a means of conveying information throughout the entire course of the task.

The Promise of Physical Shopping Agents

It is hardly surprising that physical shopping has been neglected by the agents community. After all, until very recently there simply was no reliable way to deliver customized information to individual shoppers in remote locations. However, the explosive growth of PDAs, and their increasingly sophisticated communications capabilities promise to make them effective channels of "just in time" information to users wherever they happen to be. The present invention provides an ituitive, novel agent that supports physical shopping by exploiting the promise of this developing channel that support all phases of the shopping task and solves the foregoing problems including:

Specification of Goals

Shoppers begin by indicating at least the general category of merchandise they are interested in. Shopping agents need to enable the specification of goals at various degrees of specificity. With the present invention these goals may be refined as the task progresses.

Exploration of Product Space

Before shoppers can make a selection, they need to become educated about what is available. Shopping agents can aid in this task by presenting various classes of offerings, reviews, demonstrations, etc. The present inventive Physical shopping agent can augment this by providing shoppers with a tour of the locally available offerings.

Refinement of Preferences

As shoppers learn what is available and examine the offerings their preferences evolve. Agents need to enable shoppers to refine their preferences over time. The present invention allows the user to refine their preferences.

Identification and Comparison of Candidate Products

As shoppers begin to understand what they want and what is available they typically compile a list of candidates that will be considered more carefully. The present inventive agents supports the construction and maintenance of such lists and facilitates the comparison of candidates within the list according to various criteria.

Negotiation of Offers

The present shopping agent is not restricted to providing the shopper with information. It is possible to negotiate prices and service options with retailers.

Product Selection, Purchase and Product support

The present invention facilitates the transaction itself and can be used as a channel through which product service can be delivered.

FIG. **27B** is an illustration of the Mobile Portal platform **2710** including a Mobile Portal **2712** and an Electronic Valet **2713**. The Electronic Valet **2713** includes a supporting hardware device **2716**, such as a wireless PDA, and a Mobile Portal Thin Client standard **2714** executing on top of a Thin Client Operating System **2718**. The Mobile Portal consist of an encryption and decryption element **2720**, a Mobile Portal Server **2722**, intelligent agents **2724**, a Customer intelligence element **2726**, and a Customer database **2728**.

Thin Client is a generic term used to describe a group of rapidly emerging technologies that provide a reduction in total cost of ownership through a combination of reduced hardware costs, reduced maintenance and support costs, reduced LAN/WAN bandwidth requirements, reduced down time, improved performance and enhanced security. The term "Thin" in Thin Client refers to the (very small) size of the client operating system. In contrast, traditional PC operating systems (DOS, Windows 95, etc.) are considered "Fat" Clients due to their large size and resource requirements. Despite the fact that the Thin Client operating systems are thin, the capabilities of Thin Clients are robust. Thin Client solutions are deployed today in mission critical environments and they are providing reliable and responsive access to a myriad of applications. The Mobile Portal Thin Client **2714** is a Thin Client wherein the majority of the processing is done on the Mobile Portal Server **2722** and related third party content and service providers **2730**. The user utilizes the Mobile Portal Thin Client application **2714** to select services and review information provided by the Mobile Portal Platform **2710**. The Mobile Portal Thin Client application **2714** is made more device independent by the use of a Thin Client Operating System **2718**. The Thin Client Operating System **2718** acts as a messenger between the Mobile Portal Thin Client application **2714** and the supporting hardware **2716**. The Thin Client Operating System **2718** allows the Mobile Portal Thin Client **2714** to make function calls to the Thin Client Operating System **2718** for low level hardware operations, such as display calls and user input queries. A separate Thin Client Operating system **2718** can be developed for each hardware device **2716** used as the supporting hardware for the Electronic Valet **2713**. This allows the Mobile Portal Thin Client application **2714** to run on different supporting hardware **2716** without the need for significant low level design modification.

The Mobile Portal **2712** receives data from the Electronic Valet **2713** via a packet-switched wireless network **2732**. Information received through the packet-switched wireless network is then decoded by the encryption and decryption element **2720** of the Mobile Portal **2712**. Once the data has been decoded the Mobile Portal server **2722** utilizes intelligent agents **2724**, customer intelligence **2726**, and customer data **2728** to obtain the requested data from third party content and service providers **2730**. The Mobile Portal Server **2722** utilizes intelligent software agents to respond to customer needs. The software agents **2722** utilize customer data **2728** to determine to personalize their task to the individual user's goals, habits and preferences. The customer data **2728** is in turn routinely updated by the customer and by the customer's actions. Each time a user uses the

Mobile Portal **2712** a log is kept of the user's queries and other uses of the Mobile Portal Platform **2710**. In this way, the software agents **2724** are able to utilize the user's past habits to personalize their task.

In addition to software agents **2724**, the Mobile Portal Server **2722** utilizes customer intelligence **2726** to respond to user needs. The user may utilize data-mining and pattern recognition to find the information he desires. Again, the customer data **2728** is updated to reflect the users data-mining and pattern recognition uses. Third party content and service providers **2730** are utilized by the Mobile Portal **2712** to provide the services and information requested by the users. The third party content and service providers may be accessed through the Internet or through a Mobile Portal Extranet. The intelligent agent software **2712** search through the third party providers to determine the one most suitable for the user, taking into consideration the customer's profile contained in the customer data **2728**. In this way, the user may be less specific in their queries than they would have to be without a user profile. For example, a user can request a jacket utilizing the Mobile Portal Platform **2710**. The intelligent agents would then utilize the customer data **2728** to determine more specifically what the customer actually desired. In this case, the customer data **2728** may information that this particular user likes denim jackets as opposed to leather jackets. The intelligent agents **2724** would then search for denim jackets. Of course the user profile could be overridden by the user in order to obtain information that is contrary to what is stored in the user's profile. Some typical services provided include geographic location information, audio and visual editing, personal news & entertainment, personal shopping, personal health & safety, personal organizer, personal finance, and personal communication.

Geographic location services are typically based on information received from the integrated Global Positioning System. GPS data is combined with specific user request data to provide location specific information to the user. For example, the user may be located in San Francisco and wish obtain information on fine dining in the city. The user would request fine dining information utilizing the Electronic Valet **2713**. Location data obtained from the integrated GPS receiver would be automatically combined with the user request for fine dining, and the combined message would then be transmitted to the Mobile Portal **2712**. Based on the data received, the Mobile Portal would select the appropriate service and transmit the request, in this case fine dining in San Francisco. The Mobile Portal would then transmit the response received back to the Electronic Valet **2713**. The user is then presented with the requested information, formatted and displayed on the display device of the Electronic Valet **2713**.

Audio and visual editing services are typically based on the data received from the integrated camera and microphone. The user typically captures images utilizing the integrated digital camera. However, the user may also obtain digital images from other sources, such as scanners, e-mail, and web pages. In addition, the user typically captures sound files utilizing the integrated microphone. However, audio files may also be obtained from other sources, such as e-mail, web pages, and CDs. The image and/or audio data is combined with specific user request data to provide image and audio editing capabilities to the user. For example, the user may capture an image with the integrated digital camera, and then request to edit the image using a specific photo editor. The image captured by the integrated digital camera is then combined with the user's request for photo editing, and the combined message is then transmitted to the Mobile Portal **2712**. Based on the data received, the Mobile Portal **2712** selects the appropriate service and transmits the request, in this case image editing. The Mobile Portal then transmits the response received back to the Electronic Valet **2713**. The user is then presented with the requested information, formatted and displayed on the display device of the Electronic Valet **2713**. In this case, the user would receive a user interface for image editing. The user would then use the image editing user interface to edit the image. Changes to the image are treated as request which the Mobile Portal **2712** passes on to the image editing application, running locally or on a separate server.

### Bio-Medical Sensor Integration in Accordance with a Preferred Embodiment

One embodiment of the present invention is an Electronic Valet including integrated bio-sensors, such as pressure transducers, respiratory sensors, Volumetric Sensors, and Defibrillators.

Integrated Pressure Transducers to measure blood pressure can be of two types, invasive and noninvasive. Invasive integrated pressure transducers require the user to imbed part of the unit into the blood stream, while noninvasive integrated pressure transducers do not need access to the blood stream. Pressure transducers measure the blood pressure of the patient and report it to a receiving unit, in this case the Electronic Valet. The Electronic Valet is then able to analyze and rout the data received from the pressure transducer utilizing the Mobile Portal Thin Client and Mortal Portal Server.

Respiratory sensors, such as strain gages and volumetric sensors may also be integrated into the Electronic Valet. Strain gages worn around the chest region change impedance as the gage expands and contracts according to the expansion and contraction of the chest during breathing. Volumetric sensors sense the amount of air pressure passing through the sensor, such as when a patient breathes into the volumetric sensor. Both strain gages and volumetric sensors are able to wirelessly transmit their corresponding data to the Electronic Valet unit, thus giving the user greater freedom in their activities. As with data received from pressure transducers, data received from strain gages and volumetric sensors may be analyzed and routed utilizing the Mobile Portal Platform.

Defibrillators integrated into the Electronic Valet may be utilized to sense heart functions. Defibrillators attach to the patient utilizing a saline based gel and track heartbeats through R, T, and P waves. As with strain gages and volumetric sensors, defibrillators can wirelessly transmit data to the Electronic Valet, which then analyzes and routs the data utilizing the Mobile Portal Platform.

The above mentioned bio-sensors can be integrated individually or in combination with other sensors, such as environ-sensors, other bio-sensors, or a GPS receiver, depending on the need of the particular user. For example, an elderly user with a history of heart problems could have an Electronic Valet including an integrated Defibrillator and GPS receiver. Utilizing the Mobile Portal Platform, the user could stay up-to-date on news and information about his condition, including various food and drugs that could be harmful.

In addition, the Electronic Valet is capable of sensing problems that may occur because of the heart condition, regardless of the location of the user. While walking in the park, the user may feel chest pains, the Electronic Valet would sense that the pains are being caused by difficulties

arising because of the user's heart condition. This is accomplished utilizing the integrated defibrillator and the Electronic Valet's analysis capabilities. In this case, the data received from the integrated defibrillator will exceed predetermined safety thresholds, thus alerting the Electronic Valet that an emergency has occurred. Utilizing the Mobile Portal Platform, the Electronic Valet would then notify the appropriate emergency response unit, forward that heart data to the users physician, and notify the user's family.

In addition, the Electronic Valet forwards location coordinates, received from the integrated GPS receiver, to the emergency response unit allowing them to locate and rescue the user. After treatment at the hospital, the Mobile Portal Platform is able to coordinate the users after-care program, including tracking his diet and nutrition, as well as his exercise routine and medication.

## Supporting Code in Accordance with a Preferred Embodiment

The following code is written and executed in the Microsoft Active Server Pages environment in accordance with a preferred embodiment. It consists primarily of Microsoft Jscript with some database calls embedded in the code to query and store information in the database.

Intention-Centric Interface
Create an Intention ASP Page ("intention_create.asp")

```
<%@ LANGUAGE = "JScript" %>
Response.Buffer = true;
Response.Expires = 0;
%>
<html>
<head>
    <title>Create An Intention</title>
</head>
<body bgcolor = "#FFE9D5" style = "font-family: Arial" text ="#000000">
<%
//Define some variables
up1 = Server.CreateObject("SoftArtisans.FileUp")
intention_name = up1.Form("intention_name")
intention_desc = up1.Form("intention_desc")
//intention_name = Request.Form("intention_name")
//intention_desc = Request.Form("intention_desc")
//intention_icon = Request.Form("intention_icon")
submitted = up1.Form("submitted")
items = new Enumerator(up1.Form)
%>
<%
//Establish connection to the database
objConnection = Server.CreateObject ("ADODB.Connection")
objConnection.Open("Maelstrom")
%>
<%
//Check to see if the person hit the button end do the appropriate thing
if (submitted == "Add/Delete")
{
    flag = "false"
    //loop through all the inputs
    while) !items.atEnd( ))
    {
        i = items.item( )
        //if items are checked then delete them
        if(up1.Form(i) == "on")
        {
            objConnection.Execute("delete from user_intention where
intention_id =" + i);
            objConnection.Execute("delete from intentions where
intention_id =" + i);
            objConnection.Execute("delete from tools_to_intention where
intention_id =" + i)
            flag = "true"
        }
        items .moveNext ( )
    }
    // if items were not deleted then insert whatever is in the text field in
database
    if(flag == "false")
    {
        intention_name_short = intention_name.replace(/ /gi,"")
        objConnection.Execute("INSERT INTO intentions
(intention_name;intention_desc,intention_icon) values('" + intention_name + "', '" +
intention_desc + "', '" + intention_name_short + ".gif' + "')")
        Response.write("the intentions short name is " + intention_name_short);
        up1.SaveAs("E:development/asp_examples/"+ intention_name_short
+" .gif"
    }
}
```

-continued

Intention-Centric Interface
Create an Intention ASP Page ("intention_create.asp")

```
        // Query the database to show the most recent items.
        rsCustomersList = objConnection.Execute("SELECT * FROM intentions")
%>
<input type = "Submit" name="return_to_mcp" value="Go to Main Control Panel"
onclick="location.href='default.asp"'>
<form method="post" action="intention_create.asp" enctype="multipart/form-data" >
<TABLE border=0>
<tr><td colspan="2"><font face="Arial" size="+1"><b>Enter in a new
intention</b></font></td></tr>
<tr><td><font face="Arial">Name:</font></td><td><INPUT TYPE="text"
name="intention_name"></td></tr>
<tr><td><font face="Arial">Description:</font></td><td><TEXTAREA
name="intention_desc"></TEXTAREA></td></tr>
<tr><td><font face="Arial">icon Image:</font></td><td><INPUT TYPE="file"
NAME="intention_icon" size=40></td></tr>
<tr><td colspan="2"><INPUT type="submit" name="submitted"
value="Add/Delete"></td></tr>
</TABLE>
<HR>
<font face="Arial"size="+1"><b>Current Intentions</b></font>
<TABLE>
    <tr bgcolor=E69780 align="center">
    <td>
        <FONT color="white">Delete</FONT>
    </td>
    <TD>
        <FONT color="white">Intention</FONT>
    </TD>
    <TD>
        <FONT color="white">Description</FONT>
    </TD>
    </tr>
<%
//Loop over the intentions in the list
counter = 0;
while (!rsCustomersList.EOF)
{
%>
    <tr bgcolor="white" style="font-size:smaller">
        <td align=center>
            <INPUT type="checkbox"
name="<%rsCustomerList("intention_id")%>">
        </TD>
        <td>
            <%= rsCustomersList("intention_name")%>
        </td>
        <td>
            <%= rsCustomersList("intention_desc")%>
        </td>
        <td>
            <img src=". ./images/<%= rsCustomersList("intention_icon")%>">
        </td>
<%
counter++
rsCustomersList.MoveNext( )}
%>
</TABLE>
<hr>
Available Tools
</form>
</BODY>
</HTML>
```

Retrieve Intentions List ASP Page ("intentions_list.asp")

```
<!-- #include file="include/check_authentication.inc" -->
<HTSL>
<HEAD>
    <TITLE>mySit! Intentions List</TITLE>
<SCRIPT LANGUAGE="JavaScript">
    function intentionsList ( ) {
```

-continued

Retrieve Intentions List ASP Page ("intentions_list.asp")

```
        this.internalArray = new Array( );
        <%
        // establish connection to the database
        objConnection = Server.CreateObject ("ADODB.Connection");
        objConnection.Open("Maelstrom");
        // create query
        intentionsQuery = objConnection.Execute("SELECT * FROM intentions
ORDER BY intention_name asc");
%>
    // write out the options
    numOptions = 0
<%
    while (!intentionsQuery.EOF)
            intentionName = intentionsQuery("intention_name");
            intentionIcon = intentionsQuery("intention_icon");
%>
        this.internalArray[<%= numOptions%>] = new Array(2);
        this.internalArray[<%= numOptions%>] [0] = "<%= intentionName
%>";
        this.internalArray<%= numOptions%>] [i] = "images/<%=
intentionIcon %>";
<%          numOptions++; intentionsQuery.moveNext( );      %>
<%              }           %>
    }
    numIntentions = <%= numOptions%>;
    intentionArray = new intentionsList( ).internalArray;
    function selectIntention ( ) {
        for (i=0;I<numIntentions;i++) {
            if (IntentionsListSelect.options[i].selected) {
                intentionNameTestField.value = intentionArray[i] [0];
                //intentionPicture.src = intentionArray[i][1];
                break;
            }
        }
    }
</SCRIPT>
</HEAD>
<BODY BGCOLOR="<%=Session{"main_background"}%>" style="font-family: Arial">
<CENTER>
<!--- <FORM NAME="intention_list"> --->
<TABLE FRAME="BOX" border=0 CELLPADDING="2" CELLSPACING="2">
<TR><TD COLSPAN="3" STYLE="font: 20pt arial" ALIGN="CENTER"><B>Add a mySite!
Intention</B></TD></TR>
<TR><TD COLSPAN="3"> </TD></TR>
<TR>
    <TD width="100"><font size="-1"Please Select An Intention You Would Like to
Add to Your List</font></TD>
    <TD colspan=2>
        <SELECT ID="IntentionsListSelect" NAME="IntentionsListSelect"
SIZE="10" style="font: 9pt Arial;" onClick="selectIntention( )">
        <%
        intentionsQuery.moveFirst( );
        for (j=0;j<numOptions;j++) { %>
            <OPTION VALUE="<%= intentionsQuery("intention_id} %>" <% if
(j == 0) { %> SELECTED <% } %>>
            <%= intentionsQuery("intention_name") %>
            <% intentionsQuery.moveNext( )
        }
        intentionsQuery.moveFirst( );
        %>
        </SELECT>
    </TD>
</TR>
<TR><TD COLSPAN="s"> </TD><TR>
<TR>
    <TD width="100"><font size="-1">Customize the Intention name</font></TD>
    <TD COLSPAN=2"><INPUT TYPE="test" NAME="intentionNameTextField"
ID="intentionNameTextField" SIZE="30" VALUE="<% intentionsQuery("intention_name")
%>"<>/TD>
<.TR>
<TR><TD COLSPAN="3" >"nbsp;</TD></TR>
<TR>
    <TD COLSPAN="3" ALIGN="CENTER">
        <INPUT TYPE="button" NAME="intentionCancelButton" VALUE="Cancel"
SIZE="10" ID="intentionOKButton"
```

-continued

Retrieve Intentions List ASP Page ("intentions_list.asp")

```
onClick="javaScript:top.opener.top.navframe.addAdIntention( );">
                                  
 
        <INPUT TYPE="button" NAME="intentionCancelButton" VALUE="Cancel"
SIZE="10" ID="intentionCancelButton" onClick="self.close( );">
    </TD>
>/TR>
</TABLE>
<!--- </FORM> 000>
</CENTER>
<% objConnection.Close( ); %>
</BODY>
</HTML>
```

Display User Intention List ASP Page (excerpted from "navigation.asp")

```
<DIV ID="intentionsList" style="position: absolute; width:210; height:95; left: 365pt;
top: -5; visibility: hidden; font-family: Arial; font-color: #000000; font: 8pt
Arial ; " >
<DIV style="position: absolute; top:7; left:7; height:78; width:210; z-index:2;
background: <%=Session("main_background")%>; border: solid 1pt #000000; padding: 3pt;
overflow: auto; alink: black; link: black;">
<body LINK="#000000" ALINK="#000000" vlink="black">
        <%
        // create query
        intentionsQuery = objConnection.Execute) "SELECT
user_intention." FROM user_intention, user_intention_to_persona WHERE
user_intention_to_persona.user_persona_id = " + Session("currentUserPersona") + "AND
user_intention_to_persona.user_intention_id = user_intention.user_intention_id" );
        numintentions = 0;
        Response.Write ("<SCRIPT>numintentions=" +
intentionsQuery.RecordCount' + "</SCRIPT><TABLE cellpadding='0' width='100%'
cellspacing='0'>");
        while (!intentionsQuery.EOF)
        {
        %>
        <TR><TD><a href="javascript:changeIntention('<%=
intentionsQuery("user_intention_id") %>', '<%=numintentions%>')"
onmouseover="mouseOverTab( )"onmouseout="mouseOutOfTab ( )"><font color="Black"
face=37 arial" size="-2"><%=intentionsQuery)"intention_custom_name")
%></font></a></TD><TD><IMG align="right" SRC="images/delete.gif" alt="Delete this
intention" onClick="confirmDelete)<%=intentionsQuery("user_intention_id")
%>)"></TD></TR>
<%numintentions++; intentionsQuery.moveNext( );
%>
        <%              }
Response.Write("<SCRIPT>numintentions="+numintentions +"</SCRIPT>");
        %>
<tr><td colspan="2"><hr></td></tr>
<TR><td colspan="2"><a href="javascript:changeIntention('add
. . . ',<%=numintentions%>);" onmouseover="mouseOverTab( )"
onmouseout="mouseOutOfTab( )"><font color="Black" face="arial" size="-2">add
. . . </font></a></td></TR>
        </table>
</body>
</DIV>
<DIV style="position: absolute; top:0; left:-5; width: 230; height:105; z-index:1;
" onmouseout="intentionlist.style.visibility='hidden'"
onmouseout="intentionlist.style.visibility= 'hidden'"
onmouseover="intentionlist.style.visibility= 'hidden'"></DIV>
</DIV>
</DIV>
```

While various embodiments have been described above, it should be understood that they have been presented by way of example only, and not limitation. Thus, the breadth and scope of a preferred embodiment should not be limited by any of the above described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.

What is claimed is:

1. A method for obtaining personal financial information on a mobile computing environment utilizing an interface support framework, comprising the acts of:

a) creating a query based in part on user input on a thin client computer;-

b) querying a network of information utilizing the interface support framework;

c) receiving a response to the query from the network of information through the interface support framework;

d) processing information in the response utilizing an application tool on the thin client computer, wherein the information in the response is filtered by the application tool based on one or more personal financial pattern templates containing information supplied by the user, and wherein information in each of the one or more personal financial templates represents a persona of the user; and

e) displaying any information from the response which has been selected based on the one or more personal financial pattern templates to the user.

**2**. A method for obtaining information on a mobile computing environment utilizing an interface support framework as recited in claim **1**, wherein a personal financial pattern template includes a plurality of patterns of words.

**3**. A method for obtaining information on a mobile computing environment utilizing an interface support framework as recited in claim **1**, wherein the interface support framework includes a mobile portal server.

**4**. A method for obtaining information on a mobile computing environment utilizing an interface support framework as recited in claim **1**, wherein the interface support framework includes an intelligent agent processor.

**5**. A method for obtaining information on a mobile computing environment utilizing an interface support framework as recited in claim **1**, wherein the interface support framework includes a customer intelligence framework that queries customer data.

**6**. A method for obtaining information on a mobile computing environment utilizing an interface support framework as recited in claim **1**, wherein the interface support framework includes a security framework for encrypting and decrypting information.

**7**. A method for obtaining information on a mobile computing environment utilizing an interface support framework as recited in claim **1**, wherein the interface support framework includes an interface for obtaining third party content.

**8**. A method for obtaining information on a mobile computing environment utilizing an interface support framework as recited in claim **1**, wherein the interface support framework includes support for mobile clients.

**9**. An apparatus that obtains personal financial information on a mobile computing environment utilizing an interface support framework, comprising:

a) a processor;

b) a memory that stores information under the control of the processor;

c) logic that creates a query based in part on user input on a thin client computer;

d) logic that queries a network of information utilizing the interface support framework;

e) logic that receives a response to the query from the network of information through the interface support framework;

f) logic that processes the information in the response utilizing an application tool on the thin client computer, wherein the information in the response is filtered by the application tool based on one or more personal financial pattern templates containing information supplied by the user, and wherein information in each of the one or more personal financial templates represents a persona of the user; and

g) logic that displays any information from the response which has been selected based on the one or more personal financial pattern templates to the user.

**10**. A computer program embodied on a computer-readable medium that obtains personal financial information on a mobile computing environment utilizing an interface support framework, comprising:

a) a code segment that creates a query based in part on user input on a thin client computer;

b) a code segment that queries a network of information utilizing the interface support framework;

c) a code segment that receives a response to the query from the network of information through the interface support framework;

d) a code segment processes the information in the response utilizing an application tool on the thin client computer, wherein the information in the response is filtered by the application tool based on one or more personal financial pattern templates containing information supplied by the user, and wherein information in each of the one or more personal financial templates represents a persona of the user; and

e) a code segment that displays any information from the response which has been selected based on the one or more personal financial pattern templates to the user.

**11**. A computer program embodied on a computer-readable medium that obtains information on a mobile computing environment utilizing an interface support framework as recited in claim **10**, wherein a personal finance pattern template includes a plurality of patterns of words.

**12**. A computer program embodied on a computer-readable medium that obtains information on a mobile computing environment utilizing an interface support framework as recited in claim **10**, wherein the interface support framework includes a mobile portal server.

**13**. A computer program embodied on a computer-readable medium that obtains information on a mobile computing environment utilizing an interface support framework as recited in claim **10**, wherein the interface support framework includes an intelligent agent processor.

**14**. A computer program embodied on a computer-readable medium that obtains information on a mobile computing environment utilizing an interface support framework as recited in claim **10**, wherein the interface support framework includes a customer intelligence framework that queries customer data.

**15**. A computer program embodied on a computer-readable medium that obtains information on a mobile computing environment utilizing an interface support framework as recited in claim **10**, wherein the interface support framework includes a security framework for encrypting and decrypting information.

**16**. A computer program embodied on a computer-readable medium that obtains information on a mobile computing environment utilizing an interface support framework as recited in claim **10**, wherein the interface support framework includes an interface for obtaining third party content.

**17**. A computer program embodied on a computer-readable medium that obtains information on a mobile computing environment utilizing an interface support framework as recited in claim **10**, wherein the interface support framework includes support for mobile clients.

**18**. A method as recited in claim **1**, wherein a personal financial pattern template is adapted for identifying words separated by punctuation, identifying full names by finding two capitalized words, parsing out time strings, and identifying continuous phrases of capitalized words as at least one of a company, a topic, and a location.

* * * * *

US006148289A

# United States Patent [19]

Virdy

[11] Patent Number: 6,148,289

[45] Date of Patent: Nov. 14, 2000

[54] **SYSTEM AND METHOD FOR GEOGRAPHICALLY ORGANIZING AND CLASSIFYING BUSINESSES ON THE WORLD-WIDE WEB**

[75] Inventor: **Ajaipal Singh Virdy**, Loudoun County, Va.

[73] Assignee: **LocalEyes Corporation**, McLean, Va.

[21] Appl. No.: **08/844,522**

[22] Filed: **Apr. 18, 1997**

**Related U.S. Application Data**

[60] Provisional application No. 60/017,548, May 10, 1996.

[51] **Int. Cl.**[7] ..... **G06F 17/60**

[52] **U.S. Cl.** ..... **705/1**; 705/26; 707/3; 707/6

[58] **Field of Search** ..... 705/1, 8, 26; 395/200.47, 395/200.48, 200.49; 704/2, 4, 5, 7; 707/1, 3, 6, 501, 502, 513

[56] **References Cited**

U.S. PATENT DOCUMENTS

| | | | |
|---|---|---|---|
| 4,586,091 | 4/1986 | Panaoussis | 360/40 |
| 4,809,081 | 2/1989 | Linehan | 358/432 |
| 5,200,993 | 4/1993 | Wheeler et al. | 379/93.02 |
| 5,384,835 | 1/1995 | Wheeler et al. | 379/93.25 |
| 5,404,510 | 4/1995 | Smith et al. | 707/2 |
| 5,412,804 | 5/1995 | Krishna | 707/2 |
| 5,452,445 | 9/1995 | Hallmark et al. | 707/2 |
| 5,485,608 | 1/1996 | Lomet et al. | 707/202 |
| 5,485,610 | 1/1996 | Gioelli et al. | 707/102 |
| 5,495,608 | 2/1996 | Antoshenkov | 707/3 |
| 5,500,929 | 3/1996 | Dickinson | 345/356 |
| 5,530,852 | 6/1996 | Meske, Jr. et al. | 709/206 |
| 5,764,906 | 6/1998 | Edelstein et al. | 395/200.49 |
| 5,778,367 | 7/1998 | Wesinger, Jr. et al. | 707/10 |
| 5,828,990 | 10/1998 | Nishino et al. | 704/2 |
| 5,878,233 | 3/1999 | Schloss | 709/225 |
| 5,878,398 | 3/1999 | Tokuda et al. | 705/8 |

FOREIGN PATENT DOCUMENTS

| | | | |
|---|---|---|---|
| 2 114 407 | 8/1983 | United Kingdom | G06F 3/1153 |
| WO 93/18484 | 9/1993 | WIPO | G06Q 9/62 |
| WO 95/08809 | 3/1995 | WIPO | G06F 17/30 |
| WO 95/09395 | 4/1995 | WIPO | G06F 9/45 |

OTHER PUBLICATIONS

Database 16 (IAC PROMT) on Diaolg, No. 6303723, "Imperative! Announces New Site for Locating Companies on the Internet", PR Newswire, 1 page, Jul. 22, 1996.

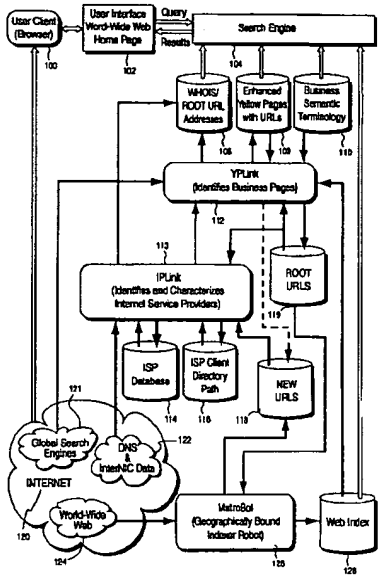
Database 148 (Trade and Industry Database) on Dialog, No. 7559252, "Open Market Inc. offers Internet Users Free Access to Directory of Commercial Sites on the Internet", Business Wire, 2 pages, Nov. 8, 1994.

*Primary Examiner*—Stephen R. Tkacs
*Assistant Examiner*—Alexander Kalinowski
*Attorney, Agent, or Firm*—Nixon & Vanderhye P.C.

[57] **ABSTRACT**

A method and search engine for classifying a source publishing a document on a portion of a network, includes steps of electronically receiving a document, based on the document, determining a source which published the document, and assigning a code to the document based on whether data associated with the document published by the source matches with data contained in a database. An intelligent geographic- and business topic-specific resource discovery system facilitates local commerce on the World-Wide Web and also reduces search time by accurately isolating information for end-users. Distinguishing and classifying business pages on the Web by business categories using Standard Industrial Classification (SIC) codes is achieved through an automatic iterative process.

**18 Claims, 6 Drawing Sheets**



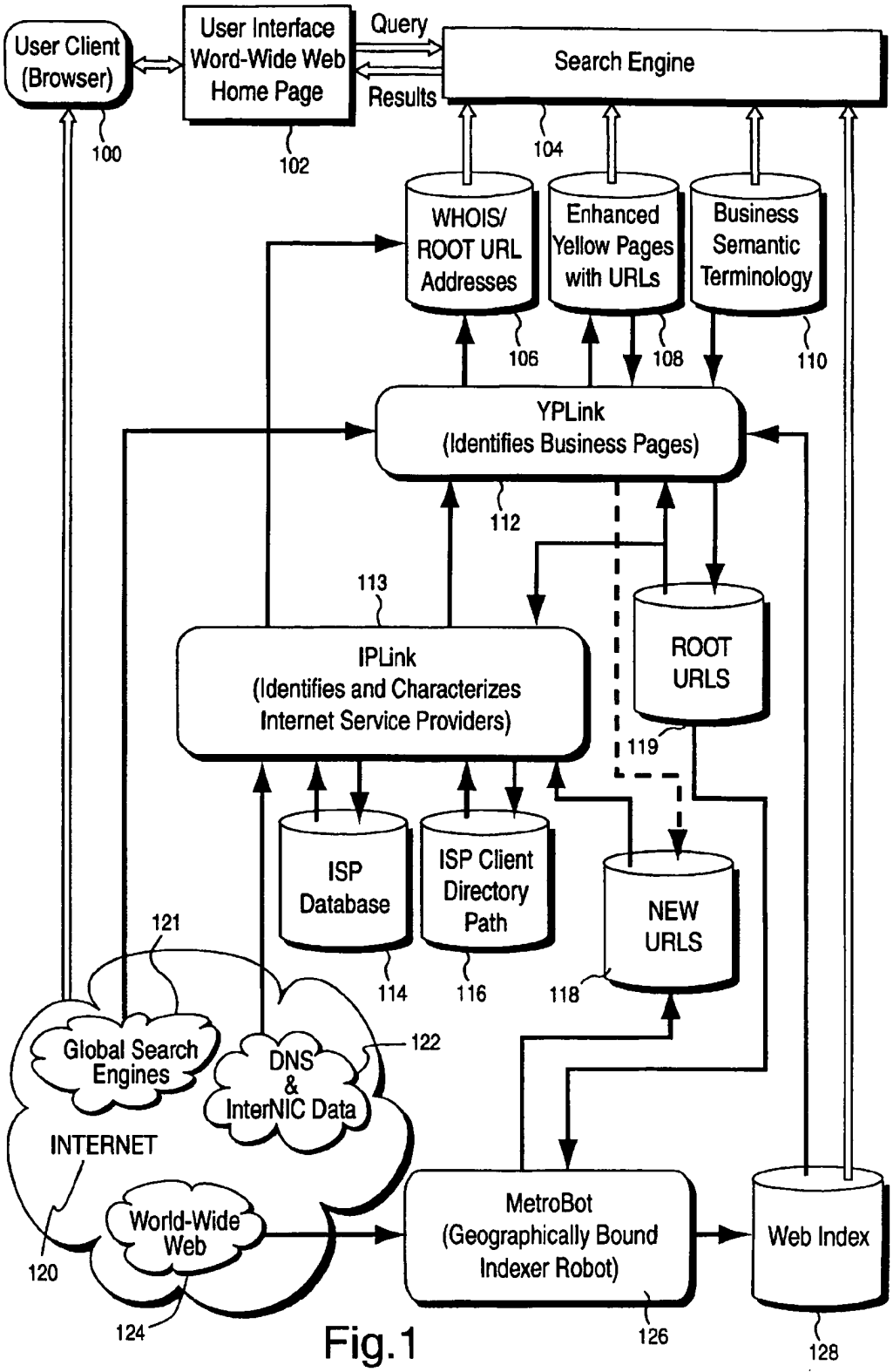

User Client (Browser)
User Interface Word-Wide Web Home Page
Query
Results
Search Engine
100
102
104
WHOIS/ ROOT URL Addresses
Enhanced Yellow Pages with URLs
Business Semantic Terminology
106
108
110
YPLink (Identifies Business Pages)
112
113
IPLink (Identifies and Characterizes Internet Service Providers)
ROOT URLS
119
ISP Database
ISP Client Directory Path
NEW URLS
114
116
118
121
122
Global Search Engines
DNS & InterNIC Data
INTERNET
World-Wide Web
120
124
MetroBot (Geographically Bound Indexer Robot)
Web Index
126
128

Fig.1

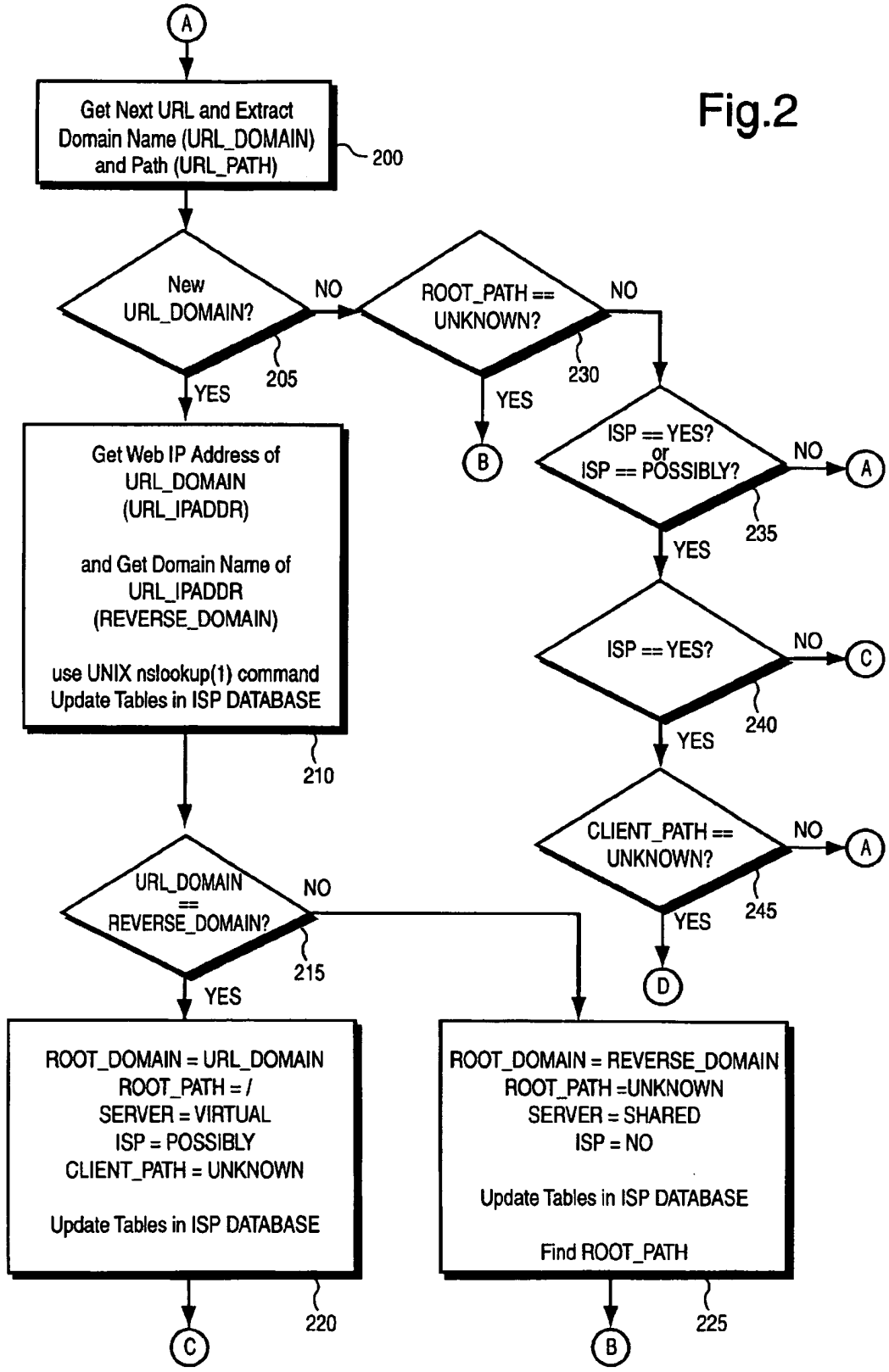

A
Get Next URL and Extract Domain Name (URL_DOMAIN) and Path (URL_PATH)
200
New URL_DOMAIN?
NO
205
YES
ROOT_PATH == UNKNOWN?
NO
230
YES
B
Get Web IP Address of URL_DOMAIN (URL_IPADDR)
and Get Domain Name of URL_IPADDR (REVERSE_DOMAIN)
use UNIX nslookup(1) command
Update Tables in ISP DATABASE
210
ISP == YES? or ISP == POSSIBLY?
NO
A
235
YES
ISP == YES?
NO
C
240
YES
CLIENT_PATH == UNKNOWN?
NO
A
245
YES
D
URL_DOMAIN == REVERSE_DOMAIN?
NO
215
YES
ROOT_DOMAIN = URL_DOMAIN
ROOT_PATH = /
SERVER = VIRTUAL
ISP = POSSIBLY
CLIENT_PATH = UNKNOWN
Update Tables in ISP DATABASE
220
C
ROOT_DOMAIN = REVERSE_DOMAIN
ROOT_PATH =UNKNOWN
SERVER = SHARED
ISP = NO
Update Tables in ISP DATABASE
Find ROOT_PATH
225
B

Fig.2

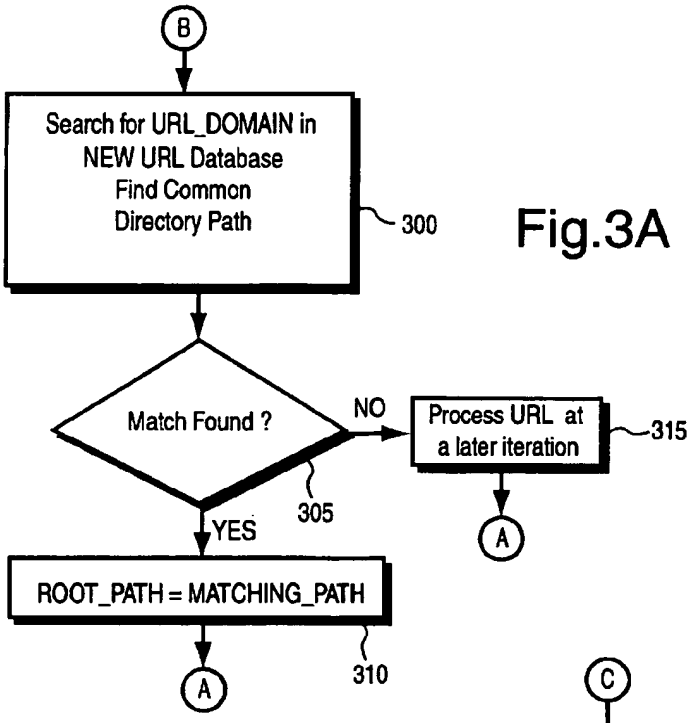
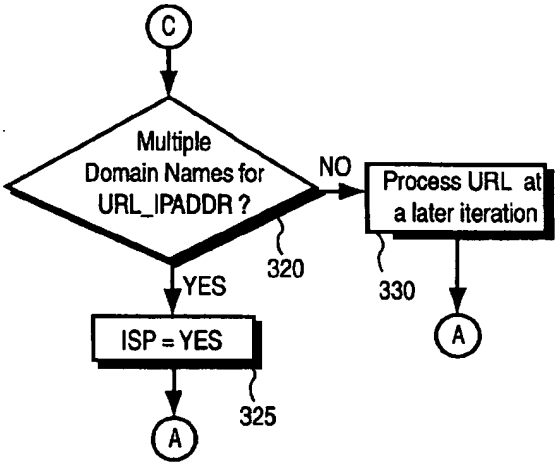
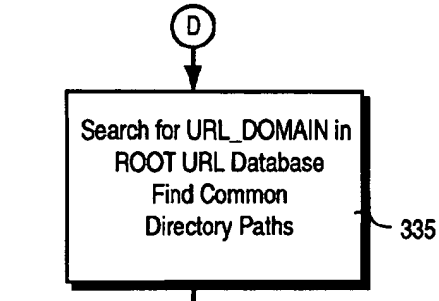
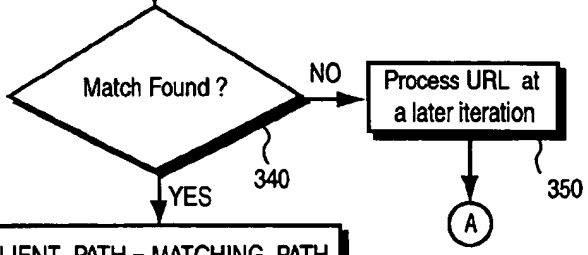



Fig.3A



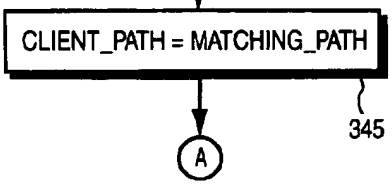
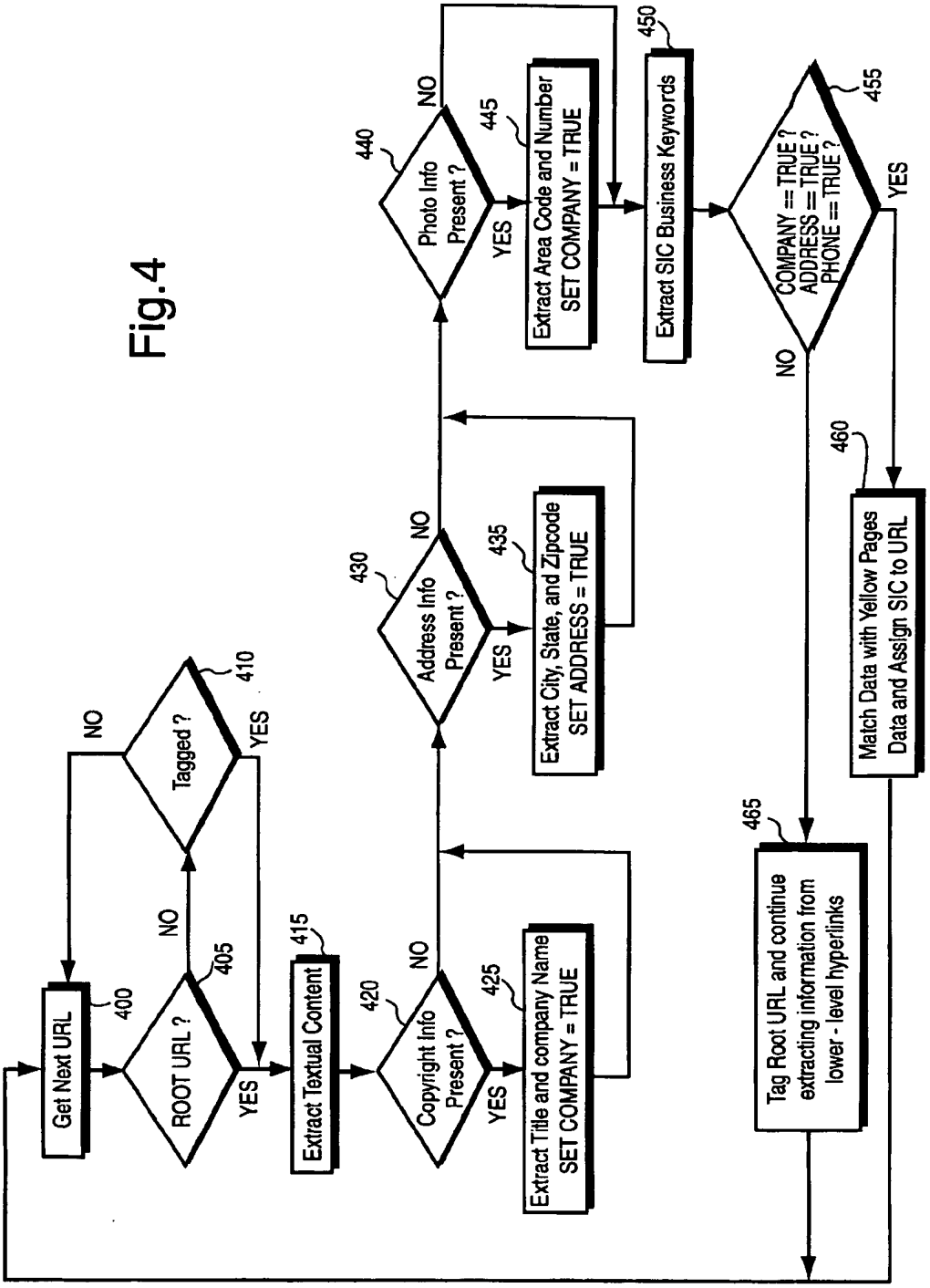




Fig.3B

Fig.3C

Fig.4


Get Next URL
400
ROOT URL ?
405
NO
YES
Tagged ?
410
NO
YES
Extract Textual Content
415
Copyright Info Present ?
420
NO
YES
Extract Title and company Name SET COMPANY = TRUE
425
Address Info Present ?
430
NO
YES
Extract City, State, and Zipcode SET ADDRESS = TRUE
435
Photo Info Present ?
440
NO
YES
Extract Area Code and Number SET COMPANY = TRUE
445
Extract SIC Business Keywords
450
COMPANY == TRUE ? ADDRESS == TRUE ? PHONE == TRUE ?
455
NO
YES
Match Data with Yellow Pages Data and Assign SIC to URL
460
Tag Root URL and continue extracting information from lower - level hyperlinks
465

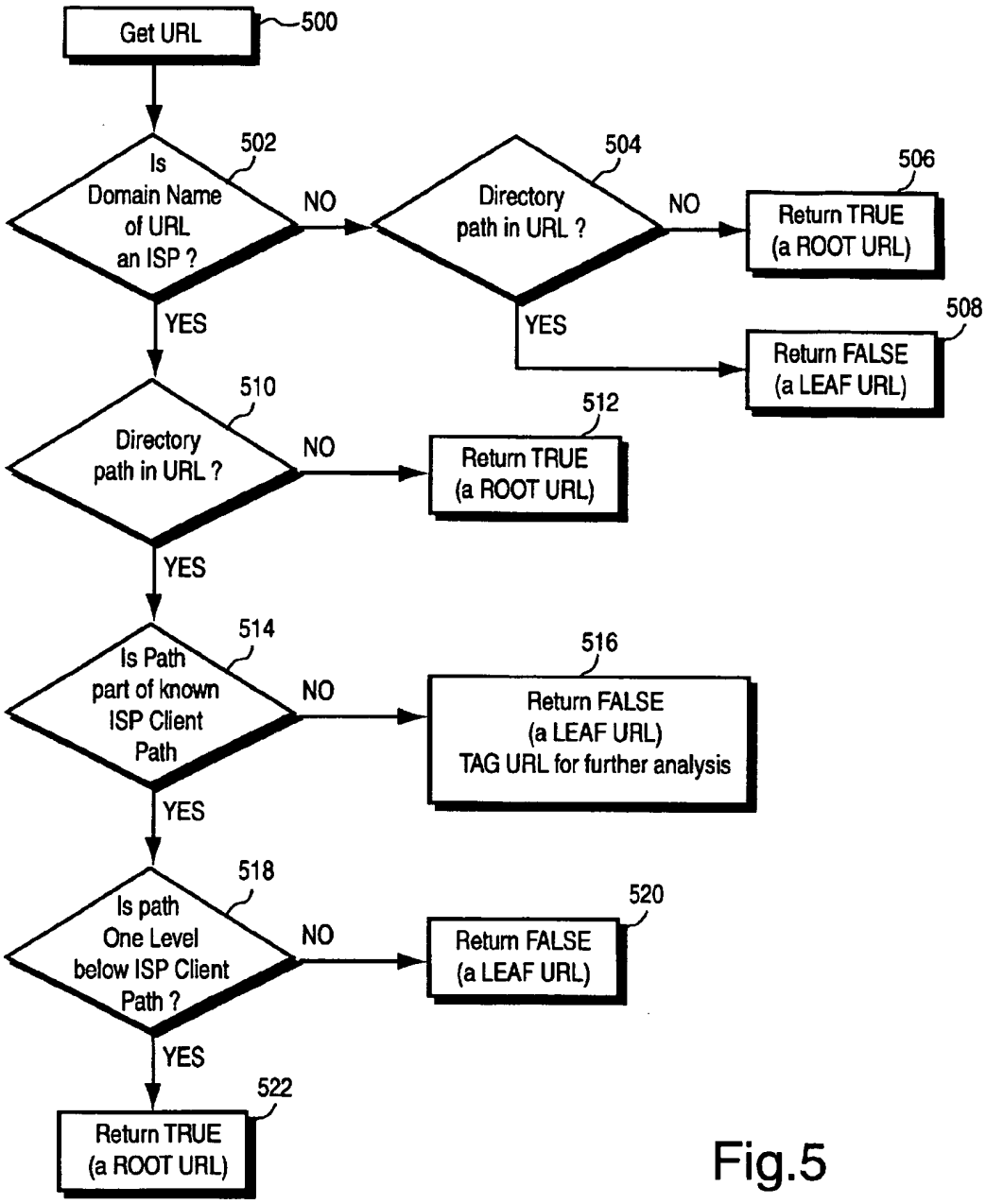

Get URL
500
Is Domain Name of URL an ISP ?
502
NO
Directory path in URL ?
504
NO
Return TRUE (a ROOT URL)
506
YES
YES
Return FALSE (a LEAF URL)
508
Directory path in URL ?
510
NO
Return TRUE (a ROOT URL)
512
YES
Is Path part of known ISP Client Path
514
NO
Return FALSE (a LEAF URL) TAG URL for further analysis
516
YES
Is path One Level below ISP Client Path ?
518
NO
Return FALSE (a LEAF URL)
520
YES
Return TRUE (a ROOT URL)
522

Fig.5

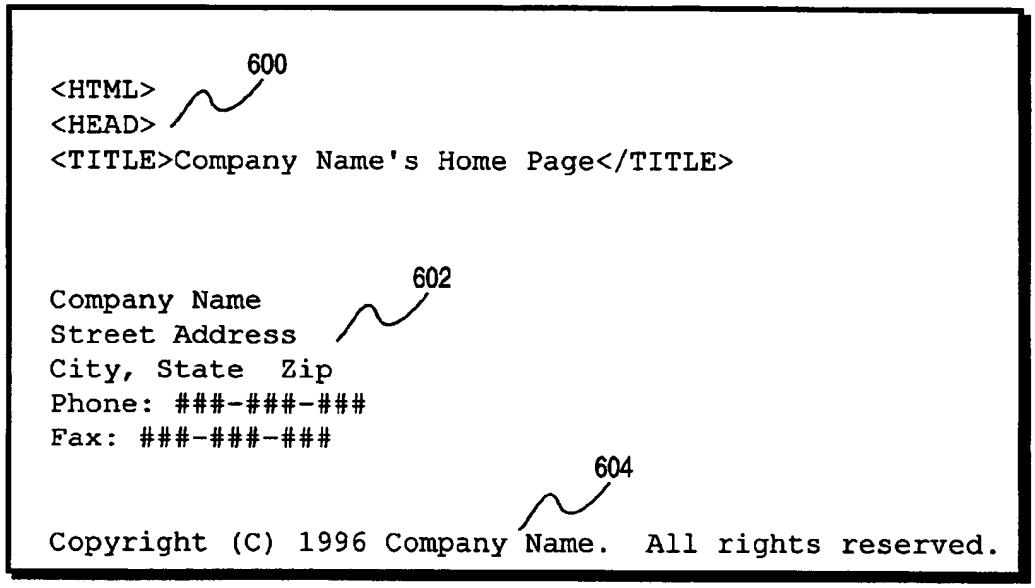

600
<HTML>
<HEAD>
<TITLE>Company Name's Home Page</TITLE>
602
Company Name
Street Address
City, State Zip
Phone: ###-###-###
Fax: ###-###-###
604
Copyright (C) 1996 Company Name. All rights reserved.

Fig.6

# SYSTEM AND METHOD FOR GEOGRAPHICALLY ORGANIZING AND CLASSIFYING BUSINESSES ON THE WORLD-WIDE WEB

This application claims priority under 35 U.S.C. Section 119 based on U.S. application Ser. No. 60/017,548, filed May 10, 1996.

## BACKGROUND OF THE INVENTION

The present invention generally relates to a resource discovery system and method for facilitating local commerce on the World-Wide Web and for reducing search time by accurately isolating information for end-users. For example, distinguishing and classifying business pages on the Web by business categories using the Standard Industrial Classification (SIC) codes is achieved through an automatic iterative process which effectively localizes the Web.

## DESCRIPTION OF THE RELATED ART

Resource discovery systems have been widely studied and deployed to collect and index textual content contained on the World-Wide Web. However, as the volume of accessible information continues to grow, it becomes increasingly difficult to index and locate relevant information. Moreover, global flat file indexes become less useful as the information space grows causing user queries to match too much information.

Leading organizations are attempting to classify and organize all of Web space in some manner. The most notable example is Yahoo, Inc. which manually categorizes Web sites under fourteen broad headings and 20,000 different sub-headings. Still others are using advanced information retrieval and mathematical techniques to automatically bring order out of chaos on the Web.

Solutions to solve this information overload problem have been addressed by C. Mic Bowman et al. using Harvest: A Scalable, Customizable Resource Discovery and Access System. Harvest supports resource discovery through topic-specific content indexing made possible by a very efficient distributed information gathering architecture. However, these topic specific brokers require manual construction and they are geared more for academic and scientific research than commercial applications.

Cornell's SMART engine developed by Gerard Salton uses a thesaurus to automatically expand a user's search and capture more documents. Individual, Inc. uses this system to sift through vast amounts of textual data from news sources by filtering, capturing, and ranking articles and documents based on news industry classification.

The latest attempts for automated topic-specific indexing include the Excite, Inc. search engine which uses statistical techniques to build a self-organizing classification scheme. Excite Inc.'s implementation is based on a modification of the popular inverted word indexing technique which takes into account concepts (i.e., synonymy and homonymy) and analyzes words that frequently occur together. Oracle has developed a system called ConText to automatically classify documents under a nine-level hierarchy that identifies a quarter-million different concepts by understanding the written English language. ConText analyzes a document and then decides which of the concepts best describe the document's topic.

The systems described above all attempt to organize the vast amounts of data residing on the Web. However, these



mathematical information retrieval techniques for classifying documents only work when the message of a document is directly correlated to the words it contains. Attempts to isolate documents by regions or to separate business content from personal content in an automated fashion is not addressed by any conventional system or structure.

## SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a method and system for overcoming the above-mentioned problems of the conventional methods and techniques.

The invention is based on a heuristic algorithm which exploits common Web page design principles. The key challenge is to ascertain the owner of a Web page through an iterative process. Knowing the owner of a Web page helps identify the nature of the content business or personal which, in turn, helps identify the geographic location.

In a first aspect of the invention, a method of classifying a source publishing a document on a portion of a network, includes steps of electronically receiving a document, based on the document, determining a source which published the document, and assigning a code to the document based on whether data associated with the document published by the source matches with data contained in a database.

In a second aspect, a search engine is provided for use on a network for distinguishing between business web pages and personal web pages. The search engine includes a mechanism for parsing the content of a hyper-text markup language (HTML) at a web address and searching for criteria contained therein, a mechanism for analyzing a uniform resources locator (URL) of the web address to determine characteristics thereof of a web page at the web address, a mechanism for determining whether the criteria match with data contained in a database, and a mechanism for cross-referencing a match, determined by the determining mechanism, to a second database, to classify a source which published the web page.

## BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, aspects and advantages will be better understood from the following detailed description of a preferred embodiment of the invention with reference to the drawings, in which:

FIG. 1 shows the process flow diagram of a geographically bound resource discovery system including three main components of the invention (sometime referred to below as "MetroSearch") identified as MetroBot, IPLink, and YPLink;

FIG. 2 depicts the IPLink flow chart, the process for identifying ISPs and Client Directory Paths;

FIGS. 3A–3C are sub-processes of the IPLink flow chart shown in FIG. 2;

FIG. 4 depicts the flow chart of YPLink for identifying business pages;

FIG. 5 is a flow diagram for determining if a given uniform resources locator (URL) is a Root URL or a Leaf URL; and

FIG. 6 is a template of a typical business home page.

## DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT OF THE INVENTION

Referring now to the drawings, and more particularly to FIG. 1, there is shown the general arrangement of a preferred embodiment according to the present invention.

The underlying insight behind the invention is that individuals and organizations responsible for the design, creation, and maintenance of their home page generally follow some basic unwritten rules. These rules can be exploited to automatically identify the owner of the home page with a high probability of success. Once the owner of the home page is determined, an SIC code is assigned to it by looking up the owner in a Yellow Pages database. If a matching entry exists, then the owner is a business, otherwise the owner is deemed to be an individual with a personal home page.

FIG. **1** shows a preferred architecture for implementing a geographically bound resource discovery system. The main components of interest are MetroBot **126**, IPLink **113**, and YPLink **112**.

The World-Wide Web ("the Web") **124** is based on a client-server architecture. The Web is the graphical, multimedia portion of the Internet **120**. The client side program is a Web browser **100** and the server side is a computer running the HTTPD program **102**. The Web server is accessed through the Internet by specifying a Uniform Resource Locator (URL). User-entered queries are sent to a back-end processor or search engine **104** which gathers results from various databases **106, 108, 110**, and **128**, and formats the request and presents them back to the user.

MetroBot **126** is an indexer robot which traverses hyperlinks in HTML documents and indexes the content into a searchable Web index database **128**. These hyperlinks or URLs point to other Web pages making it possible to recursively traverse large portions of the Web from a single, well-chosen URL (seed URL). MetroBot begins its traversal from known Root URL **119** such as the home page of a local service provider (SP), such as an internet service provider (ISP). New links that are discovered are stored in New URLs database **118**. These links are processed by IPLink **113** and YPLink **112** to extract new Root URLs at which point the whole process repeats itself. Furthermore, YPLink periodically supplements its New URL list by querying global search engines **121** using strategic keywords (e.g., regional city, county, state names, zip codes, and industry specific terms).

The first level of localization is achieved by limiting URLs to registered domain names **106**. IPLink extracts domain names from the New URL database and then queries the InterNIC database **122** where records of registered domain names containing company name, contact, street address, and Internet Protocol (IP) addresses are kept. This InterNIC database can be accessed through the Unix whois (1) command. YPLink merges the InterNIC address database **108** with the Yellow Pages data **110**. This process is described in detail below.

The next level of localization is more complex since most businesses do not have their own registered domain name. Instead, they have their home page hosted on local SPs (or ISPs) or Online Service Providers (OSPs) Web Servers.

The first step in solving this problem is for IPLink **113** to characterize URLs by their IP addresses. FIGS. **2** and **3A–3C** shows the IPLink flow logic. IPLink identifies the following attributes based on the IP addresses of New URLs:

True/Virtual Web Servers vs. Shared Web Servers.

ISP vs. Non-ISP hosts.

Root Domain of URLs.

Root Path of URLs.

Client Directory Paths if host is an ISP.

A new URL is retrieved from the New URL database **200** and is parsed into the domain name and directory path portions. If it is a new domain **205**, then its Web IP address (i.e., www.domain.name) is retrieved using the Internet Domain Name Service **122**. The Unix nslookup(1) utility **210** returns an IP address given a domain name. The corresponding IP address is stored in the ISP database **114**. A reverse lookup **210** of the Web IP address is also performed to determine **215** if the given URL is hosted on a true (or virtual) Web server **220** or a shared Web server **225**. A domain name with its own unique Web IP address indicates a true or virtual Web server (non-ISP host). Multiple domain names for a single Web IP address indicates a shared Web server (ISP host).

The official domain name (Root Domain) **220** and **225** for the IP address is the domain name of the ISP (master/slave name server information returned by whois(1) can also be used to accurately identify the ISP if the Root Domain does not correspond to the ISP). Root Domain is only used for displaying URL information on search results not for further processing.

Turning to FIG. **3A**, for shared servers **225**, the Root Path is determined by searching **300** for the given domain name in the New URL database **118** and finding common directory paths **305**. If no match is found **315**, the URL will automatically be processed at a later iteration **230**, otherwise the Root Path is set to the matching path **310**.

Turning to FIG. **3B**, for virtual servers **220**, the Root Path is simply the root directory ('/'). These servers may or may not be ISPs. If multiple domain names exist for the given IP address **320**, then it is classified as an ISP **325**, otherwise it is processed at a later iteration **330, 235** and **240**. It is possible for organizations to become ISPs in the future by simply adding/hosting new domain names on their existing Web servers.

The directory path where the ISP stores its customers Web pages is called the ISP Client Directory Path **116**. This data is initially created manually for a few local ISPs (seed ISPs). This path is identified automatically **335** by searching for the given domain name in the Root URL database **119** and finding common directory paths **340**, as shown in FIG. **3C**. If no match is found **350**, then it is processed at a later iteration **245**. Matching paths **345** point to the ISPs Client Directory Path. This process improves over subsequent iterations when enough data is gathered and patterns can be recognized from a large set of ISP Web Servers.

IPLink encompasses the first phase of identifying and characterizing IP addresses. The next phase is to automatically identify businesses hosted on ISP Web servers.

FIG. **4** shows the YPLink flow chart. YPLink determines if a Web page belongs to a business or an individual. YPLink takes its input, a URL, from IPLink. FIG. **4** shows the flow diagram for the YPLink process. The first step after retrieving a URL **400** is determining if it is a "Root URL" or a "Leaf URL" **405**.

A Root URL is the entry point for an organization's or individual's home page on the World-Wide Web. A Root URL may or may not be the same as the Home page. Leaf URLs, on the other hand, are links below an organization's Root URL. Four factors are considered in determining a Root URL:

1. Is the URL hosted on a Service Provider's Web Server?
2. Is the URL on a virtual Web Server?
3. Does the URL contain a directory path?
4. Is the directory path a known Service Provider's Client Directory?

IPLink determines the SP Client Directory Path as described above. The ISP database **114** contains information about Client Directories for various ISPs.

FIG. **5** shows the Root URL flow logic. A given URL is retrieved **500** and parsed into two components: domain name and directory path. The domain name is analyzed to see if it is an ISP **502**. If multiple IP addresses are associated with the domain name, then the domain name is an ISP. If the domain name is not an ISP, then the directory path component is checked **504**. A missing directory path signifies a Root URL **506**, otherwise it is a Leaf URL **508**.

If the domain name is an ISP **510**, then it is also a Root URL if no directory path exists **512**. If a directory path exists **514**, then the path is compared to a list of known ISP Client Directory paths. No match **516** indicates a Leaf URL, otherwise the directory path level is analyzed **518** for final Root URL determination. If the path is one directory level below the Client Directory path then it is a Root URL **522**, otherwise it is a Leaf URL **520**.

After a URL is determined to be Root URL, then the home page it points to is analyzed **415** to see if it follows some basic guidelines. A typical home page layout is illustrated in FIG. **6**. Other than following HTML requirements, there is no rule or standards for the layout of textual content. The key pieces of information required to ascertain the owner of a Web page are 1) company name, 2) zip code, and 3) telephone number. These three pieces of information do not have to exist in the Root URL. They can reside anywhere among various Leaf URLs beneath a Root URL. In many cases, this information is stored in a file called about.html. However, the same information could be stored in other, similarly named files, as would be known to those skilled in the art taking the present specification as a whole. The process described below extracts this information automatically and assigns it to the Root URL being analyzed.

The company's name is usually included in the HTML TITLE tag **600**. However, the company's name could be included in other locations, as would be known to those ordinarily skilled in the art within the purview of the present specification. The layout of the address, if present, usually is in a standard recognizable format **602**. Most businesses also tend to include copyright notices near the bottom of their documents. A string search for "copyright", "©", and "©" is performed near the bottom **604** of the home page. The company name usually appears near the copyright notice. A match of the organization or individual's name in the copyright field **420** and the TITLE field **425** provides the first indication of the owner of the home page. If no match is found, then the URL is tagged for further analysis during the next iteration.

The next step is to analyze the URL for address **430** information. Addresses have an easily identifiable format. In the U.S., the format is the city name followed by a comma and then followed by the full state name or abbreviation and finally a five or nine digit zip code. However, other common formats/methods also are possible and would be known to those ordinarily skilled in this art field to locate the zip code. This string is parsed in the HTML file after stripping all tags **435**. The only information required is the 5-digit zip code since the city and state can be determined by this field alone. YPLink stores addresses associated with Root URLs and domain names in an address database **106**.

If a phone format field is present then it is also extracted and stored **440**. U.S. phone field is a 10-digit field where the first three digits representing the area code are optionally enclosed in parentheses or separated by a dash, space, or a period, and then followed by a 7-digit number which is separated by a dash, space, or a period after the third digit **445**. Other similar methods of identifying a phone number are known to those ordinarily skilled in the art.

The pair consisting of the company name and zip code are usually enough to identify a business **455**. A query is constructed using this pair and sent to a Yellow Pages database server. This database is indexed by business names and zip codes. If a single match is found, then the resulting SIC code is assigned to the corresponding Root URL **460**. If multiple entries are matched, then the phone field is also included in the query to assure that only a single entry is retrieved. If no match is found, then the URL is tagged **465** for further analysis of lower-level hyperlinks during the next iteration. The matching data is stored in an enhanced Yellow Pages database **108**.

If no match is found at any level, then the page is tagged **450** as a personal page with an SIC code assigned according to the closest match based on the Business Semantic Terminology database **110**. This database is a proprietary thesauri of keywords relating business categories in the Yellow Pages and other emerging industries such as Internet technology to extended SIC codes.

While the invention has been described in terms of a single preferred embodiment, those skilled in the art will recognize that the invention can be practiced with modification within the spirit and scope of the appended claims.

For example, while the invention above has been described primarily in terms of (e.g., implemented in) a software process and a system employing software and hardware, the invention could also be implemented with hardware as would be known by one of ordinary skill in the art taking the present specification as a whole.

What is claimed is:

**1**. A method of classifying a document published by a source on a portion of a network, comprising the steps of:

electronically receiving a document;

based on the document, determining a source which published the document; and

assigning a code to said document based on whether data associated with the document published by the source matches with data contained in a database,

wherein said portion of said network comprises a graphical multimedia portion of said network, said source comprises a Web site publishing a home page, and said network comprises the Internet.

**2**. The method according to claim **1**, wherein said database comprises a Yellow Pages database.

**3**. The method according to claim **1**, wherein said graphical multimedia portion of said network comprises the World-Wide Web (WWW) and said document comprises a Web document, and

wherein said step of assigning a code comprises assigning a code that classifies the Web document as a first Web document type when there is a match of data associated with the Web document published by the Web site with said data contained in said database, and that classifies the Web document as a second Web document type when there is no match of data associated with the Web document published by the Web site with said data contained in the database.

**4**. The method according to claim **3**, wherein said database comprises a Yellow Pages database.

**5**. The method according to claim **3**, wherein the first Web document type is a business document and the second Web document type is a personal document.

**6**. A method of classifying a document published by a source on a portion of a network, comprising the steps of:

electronically receiving a document;

based on the document, determining a source which published the document; and

assigning a code to said document based on whether data associated with the document published by the source matches with data contained in a database,

wherein said step of determining a source includes:

extracting a domain name from a predetermined uniform resources locator (URL) database;

querying a registered domain name database for storing registered domain names; and

merging addresses from said registered domain name database with predetermined data.

**7**. The method according to claim **6**, wherein said predetermined data comprises Yellow Pages data.

**8**. The method according to claim **6**, wherein said step of determining further comprises:

parsing URLs from the predetermined URL database into domain name and directory path portions; and

determining, based on the domain name, whether the URLs from the predetermined URL database are hosted on a true server or on a shared server.

**9**. The method according to claim **8**, wherein the step of determining further comprises:

attempting to determine a root path for each URL hosted on a shared server.

**10**. A search engine for use on a network for distinguishing between business web pages and personal web pages, comprising:

means for parsing the content of a hyper-text markup language (HTML) at a web address and searching for criteria contained therein;

means for analyzing a uniform resources locator (URL) of the web address to determine characteristics of a web page at the web address;

means for determining whether said criteria match with data contained in a database; and

means for cross-referencing a match, determined by said determining means, to a second database to classify a source which published the web page,

wherein said criteria include at least one of an address, a telephone number, a facsimile number, a contact and a key-word contained in said HTML, and

wherein the characteristics of said web page include a geographical location and a web page host computer.

**11**. A search engine for use on a network for distinguishing between business web pages and personal web pages, comprising:

means for parsing the content of a hyper-text markup language (HTML) at a web address and searching for criteria contained therein;

means for analyzing a uniform resources locator (URL) of the web address to determine characteristics of a web page at the web address;

means for determining whether said criteria match with data contained in a database; and

means for cross-referencing a match, determined by said determining means, to a second database to classify a source which published the web page,

wherein said second database includes a Business Semantic Terminology database having information related to business categories in a Yellow Pages directory.

**12**. A search engine for use on a network for distinguishing between business web pages and personal web pages, comprising:

means for parsing the content of a hyper-text markup language (HTML) at a web address and searching for criteria contained therein;

means for analyzing a uniform resources locator (URL) of the web address to determine characteristics of a web page at the web address;

means for determining whether said criteria match with data contained in a database; and

means for cross-referencing a match, determined by said determining means, to a second database to classify a source which published the web page,

wherein said second database includes a Yellow Pages database.

**13**. A search engine for use on a network for distinguishing between business web pages and personal web pages, comprising:

means for parsing the content of a hyper-text markup language (HTML) at a web address and searching for criteria contained therein;

means for analyzing a uniform resources locator (URL) of the web address to determine characteristics of a web pare at the web address;

means for determining whether said criteria match with data contained in a database; and

means for cross-referencing a match, determined by said determining means, to a second database to classify a source which published the web page,

wherein said web page comprises hyperlinks, and said means for parsing comprises an indexer robot for traversing said hyperlinks in said web page and a web page index database,

said indexer robot for indexing a content of said web page into said web index database.

**14**. A search engine for use on a network for distinguishing between business web pages and personal web pages, comprising:

means for parsing the content of a hyper-text markup language (HTML) at a web address and searching for criteria contained therein;

means for analyzing a uniform resources locator (URL) of the web address to determine characteristics of a web page at the web address;

means for determining whether said criteria match with data contained in a database; and

means for cross-referencing a match, determined by said determining means, to a second database to classify a source which published the web page,

wherein said means for analyzing comprises:

means for determining whether said URL comprises one of a root URL and a leaf URL.

**15**. A search engine according to claim **14**, wherein said root URL comprises an entry point for the web page on the World-Wide Web, and a leaf URL comprises a link below a root URL, said search engine further comprising:

means for parsing said URL into a domain name component and a directory path component;

means for analyzing the domain name in said domain name component to determine whether it is associated with a service provider (SP);

means for checking the directory path component to judge whether a directory path is missing, when the domain name is not associated with an SP, a missing directory path indicating a root URL, and for checking whether a directory path does not exist to thereby determine that said domain name comprises a root URL, when the domain name is associated with an SP;

means for comparing the path to known SP Client Directory paths, when a directory path exists;

means for analyzing a home page associated with said root URL, when said URL is determined to be a root URL, thereby automatically to extract home page data contained therein; and

means for assigning the home page data to the Root URL being analyzed.

**16**. A method of indexing textual content on the world-wide web, comprising:

robotically traversing the world-wide web to identify uniform resource locators; and

determining whether the identified uniform resource locators are associated with a business or an individual,

wherein the determining step comprises:

extracting ownership data from content associated with the identified uniform resource locators;

querying a business listing database based on the ownership data; and

determining that the identified uniform resource locators are associated with businesses if the querying matches the ownership data to a business listing in the business listing database.

**17**. The method according to claim **16**, further comprising:

assigning business category codes to the uniform resource locators associated with businesses.

**18**. The method according to claim **17**, wherein the business category codes are the Standard Industrial Classification (SIC) codes.

* * * * *

US005659729A

# United States Patent [19]

Nielsen

[11] Patent Number: 5,659,729

[45] Date of Patent: Aug. 19, 1997

[54] **METHOD AND SYSTEM FOR IMPLEMENTING HYPERTEXT SCROLL ATTRIBUTES**

[75] Inventor: **Jakob Nielsen**, Atherton, Calif.

[73] Assignee: **Sun Microsystems, Inc.**, Mountain View, Calif.

[21] Appl. No.: **595,477**

[22] Filed: **Feb. 1, 1996**

[51] **Int. Cl.**[6] ..... **G06F 17/30**

[52] **U.S. Cl.** ..... **395/603**; 395/610; 395/611; 395/774; 395/762; 395/793

[58] **Field of Search** ..... 395/611, 603, 395/610, 774, 762, 793

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,572,643 11/1996 Judson ..... 395/793

OTHER PUBLICATIONS

Article entitled "As We May Think", Vannevar Bush, Atlantic Monthly, Jul. 1945, reprinted with updated commentary Mar. 1996, pp. 35–67.

"A Brief History of HTML", http://www.fcs.uga.edu/ca/web_seminar/publishing/history.html, 3 Mar. 1997, pp. 1–4. Mar. 1997.

"Extensions to HTML 2.0", http://www.mcom.com/assist/net_sites/html_extensions.html, 3 Mar. 1997, pp. 1–4. Mar. 1997.

"Extensins to HTML 3.0", http://www.mcom.com/assist/net_sites/html_extensions3.html, 3 Mar. 1997, pp. 1–4. Mar. 1997.

"Publication History", http://www.bilkent.edu.tr/pub/WWW/MarkUp/HTML.html, 3 Mar. 1997, pp. 1–2. Mar. 1997.

"Paul Lintz's Homepage", http://www.erols.com/plimtz/, p1, Document Source, p. 2., 3 Mar. 1997. Mar. 1997.

"Tools for creating Web Pages.(World Wide Web Home Page Creation)", Seybold Report on Publishing Systems, vol. 24, No. 17, 1 May 1995, pp. S14–S19. May 1995.

*Primary Examiner*—Paul V. Kulik
*Assistant Examiner*—Paul K. Lintz
*Attorney, Agent, or Firm*—Timothy J. Crean

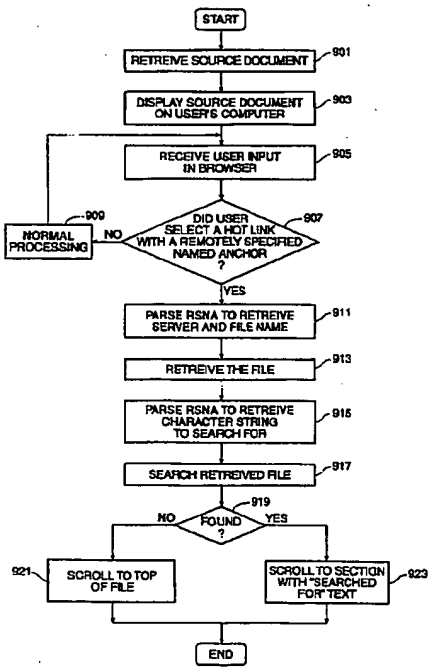
[57] **ABSTRACT**

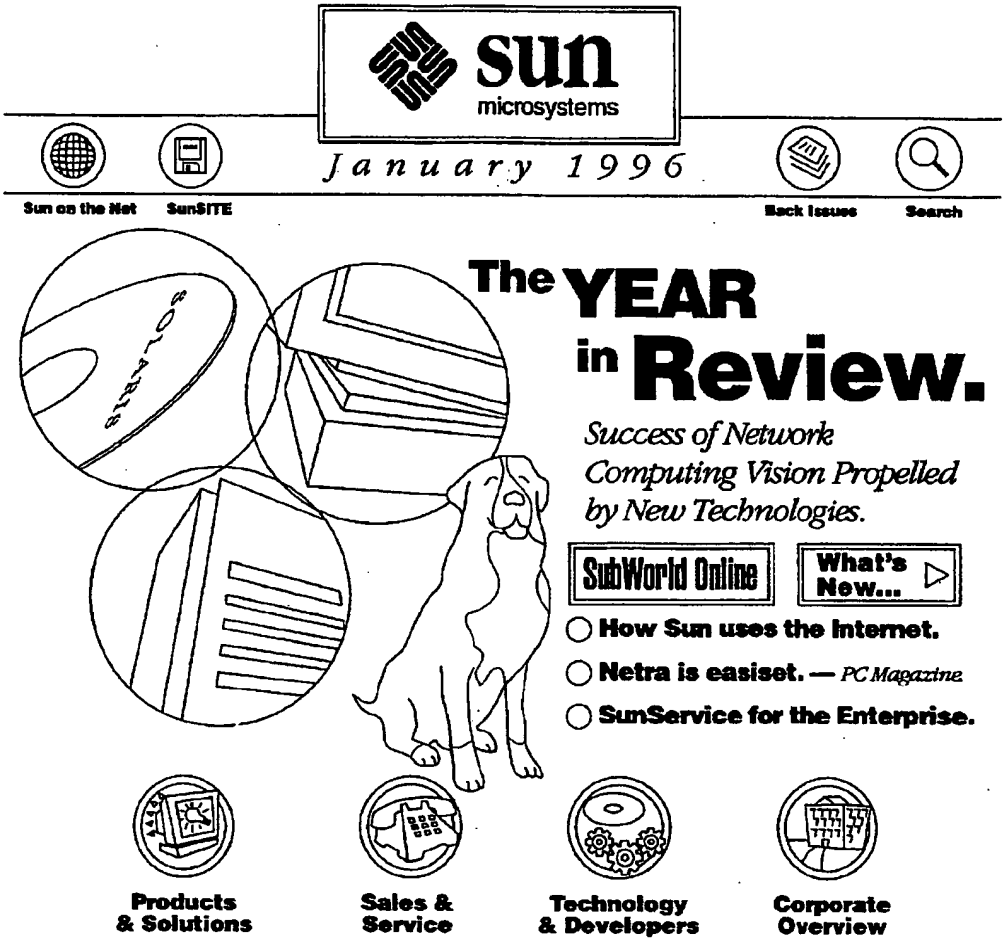
Embodiments of the present invention use a new extension to the HTML language to support remotely specified named anchors. A remotely specified named anchor, when embedded within a source document, instructs a browser program to access a portion of a destination document indicated in the remotely specified named anchor. When the browser program reads a remotely specified named anchor such as

<a href=http://foo.com/bar.html/SCROLL="Some Text">

from the source document, the browser program performs the following steps: 1) the browser retrieves the destination file "bar.html" from the server "foo.com", 2) the browser searches the file bar.html for "Some Text", and 3) if the browser finds the character swing being searched for, then the browser displays the file bar.html, scrolled to the line containing the first character of the character string being searched for.

**15 Claims, 6 Drawing Sheets**






Sun Microsystems text-only home page

Questions or comments regarding this service? *webmaster@sun.com*



FIG. 1

```
<!-- HEAD_START -->
<HTML>
<HEAD>
<TITLE>Sun Microsystems</TITLE>
<!--META NAME-"owner" VALUE="hooper@bcci.eng.sun.com" -->
</HEAD>

<BODY>
<!-- HEAD_END -->


<A HREF-"/cgi-bin/imagemap/960101/homepage.9601.map"><IMG BORDER-0 SRC-"/share/i
mages/homepage.9601.color.580x576.gif" ALT="Highly graphic homepage" ISMAP></A><
P>

Sun Microsystems <A HREF="/960101/index.textonly.html">text-only</A> home page.

<!-- FOOT_START -->
<HR>
<FONT SIZE=2> Questions or comments regarding this service?
<A HREF="/cgi-bin/comment-form.pl?/960101/index.html"><EM>webmaster@sun.com</EM>
</A></FONT>
<P>
<H5><A HREF="/share/text/SMIcopyright.html">Copyright</A> 1996 Sun Microsystems,
Inc., 2550 Gracia Ave., Mtn View, CA 94043-1100 USA. All Rights Reserved</H5>
</BODY>
</HTML>
<!-- FOOT_END -->
```

FIG. 2

```
<html>
<head>
<title>

</title>
</head>
<!--this is a comment-->
<body>

<address>

</address>
</body>
</html>
```

FIG. 3

| <start tag> | <endtag> | function |
|---|---|---|
| <html> | </html> | HTML document indicator. |
| <head> | </head> | Define document head. |
| <title> | </title> | Document title information. Should be descriptive, used in indexing and search engines. |
| <body> | </body> | Document body |
| <h (n) >, <h1>... <h6> | </h (n) >, </h1>... </h6> | Headings. h1 is largest, h6 is smallest |
| <! --- ---> | | Comment, No ending tag required |

FIG. 4

**Paragraph Formatting**

| <start tag> | <endtag> | function |
|---|---|---|
| <br> | | Break, starts a new line, no ending tag required |
| <p> | | Paragraph (break plus space), no ending tag required |
| <hr> | | Horizontal rule (horizontal line) |
| <pre> | </pre> | Preformatted text, not processed by browser. Useful for keeping spaces in tables or lists formatted in a text editor. |
| <blockquote> | </blockquote> | Blockquote |

**Character Formatting**

| <start tag> | <endtag> | Logical or Physical | HTML function |
|---|---|---|---|
|  |  | Logical | *Emphasized* |
| <var> | </var> | Logical | Variable |
| <cite> | </cite> | Logical | Citation |
|  |  | Physical | *Italics* |
|  |  | Physical | **Bold** |
| <code> | </code> | Logical | Code |
| <samp> | </samp> | Logical | Sample |
| <kbd> | </kbd> | Logical | Keyboard entry |
| <tt> | </tt> | Physical | Teletype |
| <key> | </key> | Logical | Keyword |
| <dfn> | </dfn> | Logical | Definition |
| <strike> | </strike> | Physical | ~~Strike-through~~ |
|  |  | Logical | Strong |

FIG. 5

**THOMAS JEFFERSON**

THOMAS JEFFERSON WAS ONE OF THE DRAFTERS OF THE AMERICAN CONSTITUTION.

*FIG. 6*

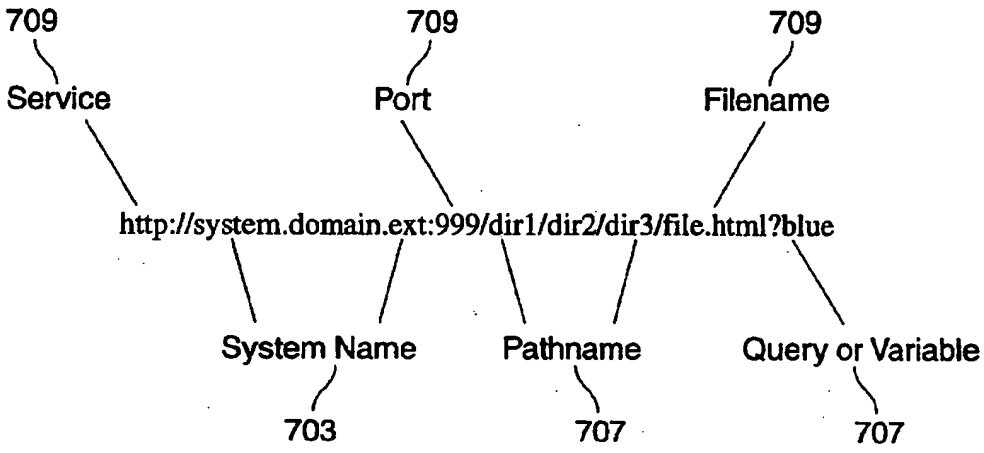



*FIG. 7*

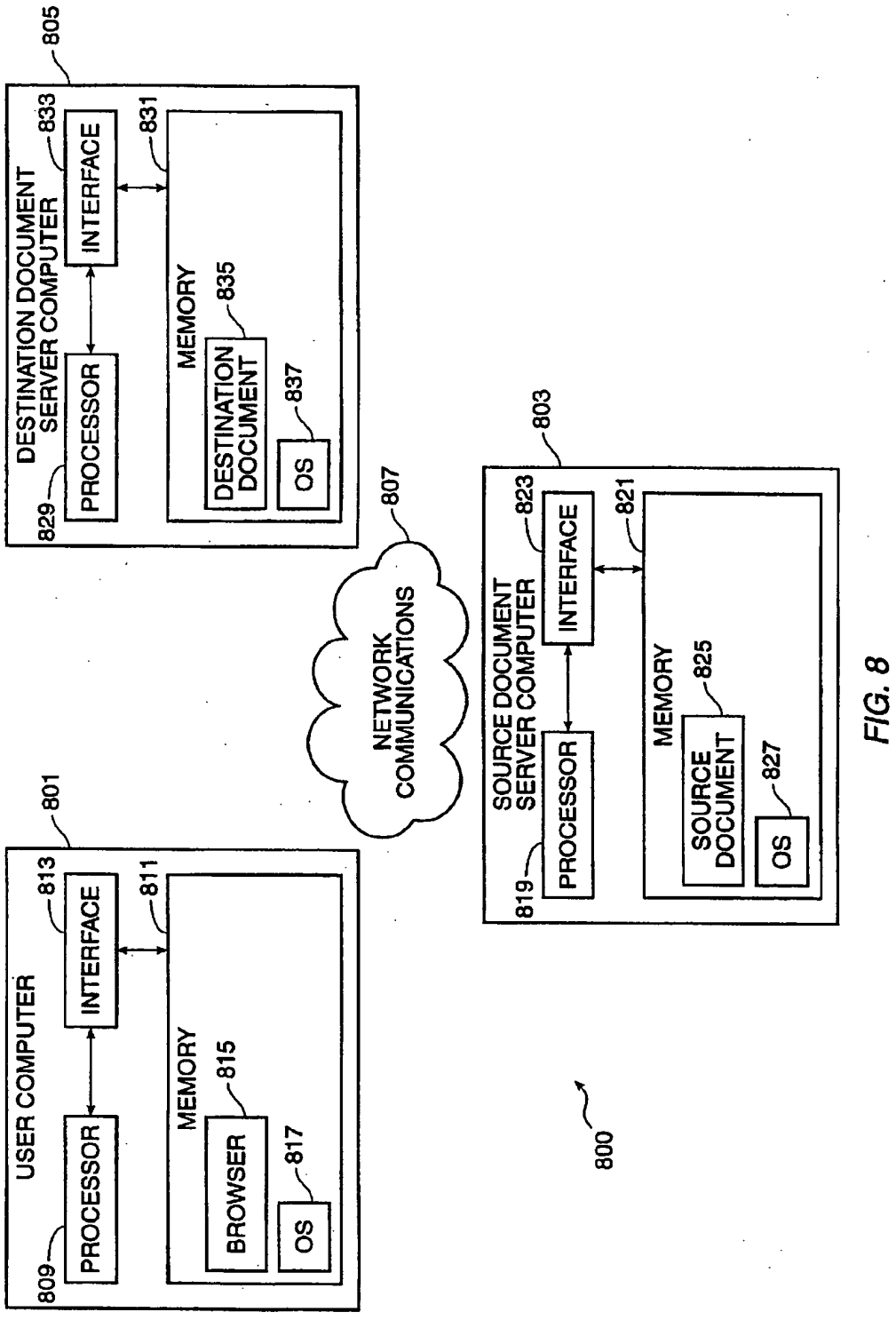

805
DESTINATION DOCUMENT SERVER COMPUTER
833
INTERFACE
831
MEMORY
835
DESTINATION DOCUMENT
837
OS
829
PROCESSOR
807
NETWORK COMMUNICATIONS
803
SOURCE DOCUMENT SERVER COMPUTER
823
INTERFACE
821
MEMORY
825
SOURCE DOCUMENT
827
OS
819
PROCESSOR
801
USER COMPUTER
813
INTERFACE
811
MEMORY
815
BROWSER
817
OS
809
PROCESSOR
800

FIG. 8

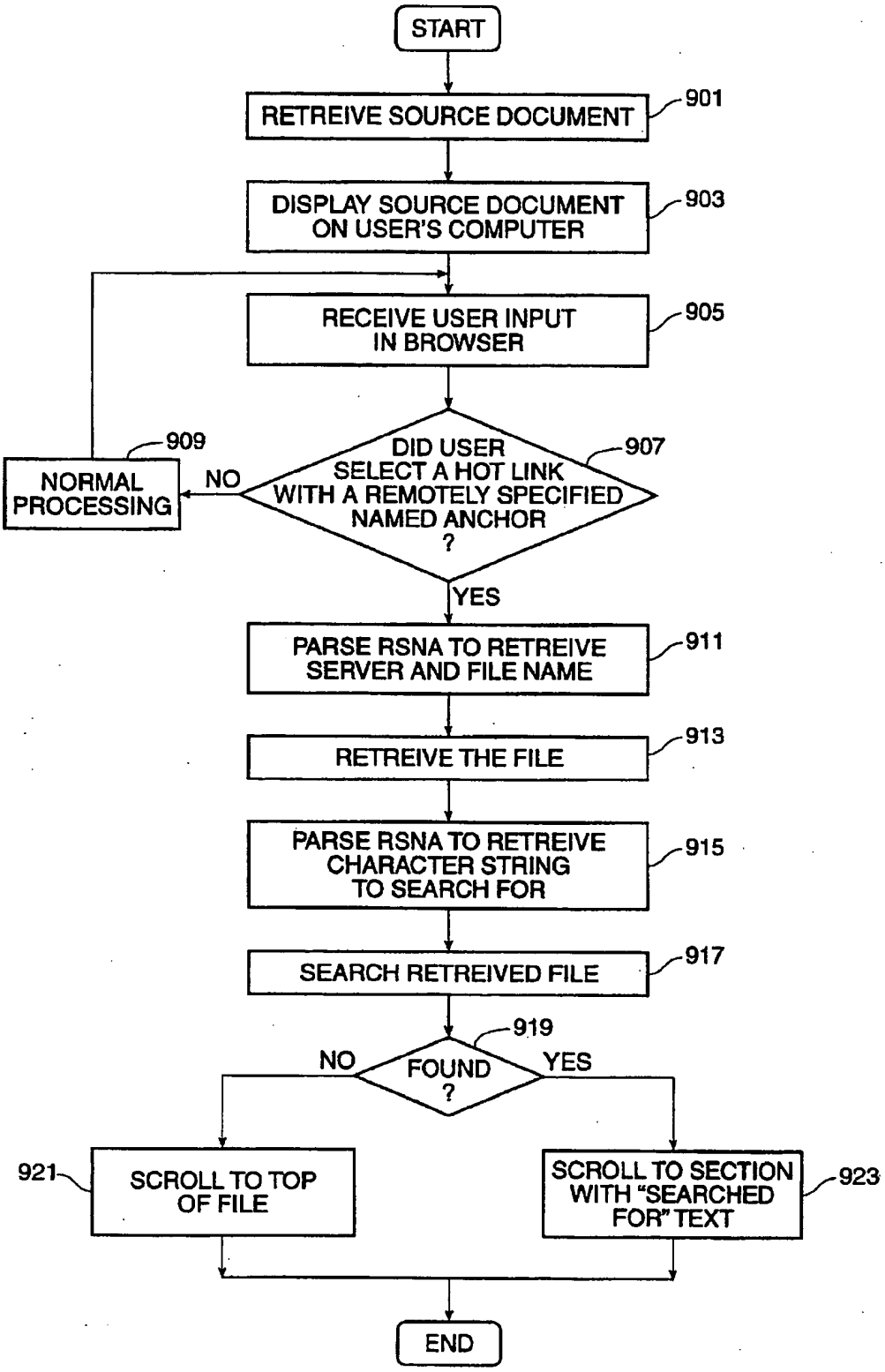

START
RETREIVE SOURCE DOCUMENT
901
DISPLAY SOURCE DOCUMENT ON USER'S COMPUTER
903
RECEIVE USER INPUT IN BROWSER
905
DID USER SELECT A HOT LINK WITH A REMOTELY SPECIFIED NAMED ANCHOR ?
907
NO
NORMAL PROCESSING
909
YES
PARSE RSNA TO RETREIVE SERVER AND FILE NAME
911
RETREIVE THE FILE
913
PARSE RSNA TO RETREIVE CHARACTER STRING TO SEARCH FOR
915
SEARCH RETREIVED FILE
917
FOUND ?
919
NO
YES
SCROLL TO TOP OF FILE
921
SCROLL TO SECTION WITH "SEARCHED FOR" TEXT
923
END

FIG. 9

# METHOD AND SYSTEM FOR IMPLEMENTING HYPERTEXT SCROLL ATTRIBUTES

## FIELD OF THE INVENTION

Aspects of the present invention provide a method and system for remotely specifying which section of a hypertext document to display on a user's computer.

## BACKGROUND OF THE INVENTION

HTML is a "markup" language which allows an author to turn a simple text document into a hypertext document for the World Wide Web ("the web "). FIG. 1 is an example of a hypertext document from Sun Microsystems as viewed through a browser from Netscape Communications, Inc. FIG. 2 illustrates the HTML source code which describes the hypertext document of FIG. 1.

The HTML markup language is analogous in some ways to the formatting codes used in word processing documents. A word processing document viewed through a word processing program is actually a combination of the text that you see and a series of hidden formatting codes (e.g., carriage return, bold, underline) which instruct the word processing program to display the word processing document in a specified way. Similarly, a hypertext document is actually a combination of the text that you see and a series of hidden "tags" or "anchors" (for new paragraphs, graphics images, hypertext links, etc.) which instruct the browser program to display the hypertext document in a specified way.

A hypertext document is usually broken down into sections, with each section delineated by one or more HTML tags. HTML tags are formatting codes surrounded by the characters < and > (less than and greater than symbols). Some HTML tags have a start tag and an end tag. In general, end tags are in the format </"symbol"> where the "symbol" is the character string found between the characters < and > in the start tag. FIG. 3 is an example of a series of HTML document tags forming a template for a typical hypertext document. For example, the document of FIG. 3 is defined as an HTML document using the tags <html> and </html>Then the "head" to the document, which typically includes a title, is defined using the tags <head>, </head>, <title>, and </title>, respectively. Following the head comes the "body" of the document which is often organized into subtopics with different levels of headings. The body is defined by the tags <body> and </body>. Headings are indicated by the tags <h#> and </h#>, where #is the level of the heading. Heading levels indicate the relative size of the heading. Heading level 1 is the largest heading size and heading level 6 is the smallest heading size. Finally, it is good practice to indicate the author of the document at the bottom of the document using the tags <address> and </address>. FIG. 4 summarizes this information in a table format.

Once the HTML template has been established, text is added to create a basic hypertext document. In order to improve readability, the author adds HTML character and paragraph formatting tags to the document. For example, the <p> tag instructs the browser to begin a new paragraph. If an author wants to highlight some text in bold, the author inserts the  tag at the beginning of the text to be highlighted and inserts a  tag at the end of the text to be highlighted. The tags  and  indicate text to display in italics. FIG. 5 illustrates additional tags for formatting characters and paragraphs.



If HTML was merely made up of the document, paragraph, and character formatting tags discussed above, it would only allow an author to define a document which stands by itself. Fortunately, additional HTML tags allow an author to "link" documents together. If a reader of a hypertext document wants to know more about a topic before reading the rest of the current hypertext document, the reader selects a "link" or "hot link", which retrieves and displays a new document that provides related information. FIG. 6 illustrates a hypertext document (i.e, a "source document") on Thomas Jefferson with a hot link named "the American Constitution". The link could take the reader to a second hypertext document (i.e., a "destination document") which, for example, displays the text of the American Constitution or which provides more information on Thomas Jefferson's role in the drafting of the American Constitution.

In HTML, a hot link to a destination document is made by placing a "reference anchor" around the text to be highlighted (e.g., "the American Constitution") and then providing a network location where the destination document is located. Reference anchors extend the idea of start and end tags. A reference anchor is created when the start tag <a> and the end tag </a> are placed around the text to be highlighted (e.g., <a> the American Constitution </a>). Then attribute information that identifies the network location of the destination document is inserted within the <a> reference tag. In HTML, the "href=" attribute, followed by the network location for the destination document, is inserted within the <a> tag. For example,

<a href="network location for the destination document"> the American Constitution </a>illustrates the basic format for a reference anchor. On the web, network locations of hypertext documents are provided using the Universal Resource Locator ("URL") naming scheme. FIG. 7 illustrates the primary components of a URL.

A service type **701** is a required part of a URL. The service type tells the user's browser how to contact the server for the requested data. The most common service type is the HyperText Transport Protocol or http. The web can handle several other services including gopher, wais, ftp, netnews, and telnet and can be extended to handle new service types. A system name **703** is also a required part of a URL. The system name is the fully qualified domain name of the server which stores the dam being requested. A port **705** is an optional part of a URL. Ports are the network socket addresses for specific protocols. By default, http connects at port **80**. Ports are only needed when the server does not communicate on the default port for that service. A directory path **707** is a required part of a URL. Once connected to the system in question, a path to the file must be specified. A filename **709** is an optional part of a URL. The file name is the data file itself. The server can be configured so that if a filename isn't specified, a default file or directory listing is returned. A search component **711** is another optional part of a URL. If the URL is a request to search a data base, the query can be embedded in the URL. The search component is the text after the ? or #in a URL.

Substituting the URL "http://system/dir/file.html" into the example above, the reference anchor:

<a href="http://system/dir/file.html/"> the America Constitution </a>

identifies an html file to retrieve and display when a user selects "fie American Constitution" hot link.

Sometimes an author may want to direct the reader's attention not to the destination document as a whole but to

a specific part of the destination document. For example, instead of pointing the reader to the beginning (i.e., the Preamble) of the American Constitution, an author may want to point the reader directly to the 10th Amendment (i.e., Article X) of the American Constitution. Hypertext links that point to a specific point in a destination document are known as named anchors. Named anchors are essentially modified reference anchors. Continuing with the example above, if an author wants to point to the section on the 10th Amendment within a destination document containing HTML source code for the entire American Constitution, then the author follows a two step process. First the author modifies the HTML source code for the destination document by inserting a "NAME" attribute within an <a> tag which is inserted before the start of the section on the 10th Amendment. For example, the tag

```
<a NAME="10th Amendment"> Article X </a>
```

could be inserted into the destination document's HTML, source code before the start of the section on Article X. To reference this point, the author of the source document creates a named anchor in the source document which uses a #character to reference the"10th Amendment" NAME attribute in the destination document. For example, the named anchor:

```
<a href="http://system/dir/file.html#10th Amendment"> the 10th
Amendment </a>
```

identifies the section on Article X as the section to retrieve and display when a user selects the hot link "the 10th Amendment".

An implicit assumption of the example set forth above is that the author of the source document has permission to edit and modify the destination document in order to add a "NAME" attribute before the section on Article X. At the very least, the author of the source document has to be able to convince the author of the destination document to add such a "NAME" attribute before the section on Article X. However, since the web is a distributed, network-based hypertext system, the author of the source document may in fact not have access to the destination document. Thus, it would be beneficial to provide a method and system which allows browsers to automatically display sections of destination documents, even though those sections do not include embedded NAME attributes.

## SUMMARY OF THE INVENTION

Embodiments of the present invention use a new extension to the HTML language to support remotely specified named anchors. A remotely specified named anchor, when embedded within a source document, instructs a browser program to access a portion of a destination document indicated in the remotely specified named anchor. One benefit of the present invention over previously implemented named anchors is that embodiments of the present invention provide this functionality even when the indicated portion of the destination document does not contain a "NAME" attribute. In this way, an author of a source document can create a hot link which scrolls to an indicated portion of a destination document even though the author of the source document is unable to modify, or have modified, the source code of the destination document to include a "NAME" attribute.

In one embodiment, when the browser program reads a remotely specified named anchor such as:

```
<a href=http://foo.com/bar.html SCROLL="Some Text">
```

the browser program performs the following steps: 1) the browser retrieves the file "bar.html" from the server "foo.com",2) the browser searches the file bar.html for "Some Text", and 3) if the browser finds the character string being searched for, then the browser displays the file bar-.html scrolled to the line containing the first character of the character string being searched for.

The present invention also provides graceful degradation to support legacy browsers. If a remotely specified named anchor such as:

```
<a href=http://foo.com/bar.html SCROLL="Some Text">
```

is read by a browser program which does not support the new HTML extension of the present invention then the legacy browser will simply ignore the SCROLL attribute and will instead display the destination file bar.html in the normal fashion, i.e., scrolled to the top of the file.

## NOTATIONS AND NOMENCLATURE

The detailed descriptions which follow are presented largely in terms of methods and symbolic representations of operations on data bits within a computer. These method descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art.

A method is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. These steps require physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It proves convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like. It should be bourne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities.

Useful machines for performing the operations of the present invention include general purpose digital computers or similar devices. The general purpose computer may be selectively activated or reconfigured by a computer program stored in the computer. A special purpose computer may also be used to perform the operations of the present invention. In short, use of the methods described and suggested herein is not limited to a particular computer configuration.

## BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is an example of a hypertext document from Sun Microsystems as viewed through a browser from Netscape Communications, Inc.

FIG. 2 illustrates HTML source code which describes the hypertext document of FIG. 1.

FIG. 3 is an example of a series of HTML document tags forming a template for a typical hypertext document.

FIG. 4 summarizes information regarding HTML document tags.

FIG. 5 summarizes information regarding HTML character and paragraph tags.

FIG. 6 illustrates a hypertext document on Thomas Jefferson with a hot link named "the American Constitution".

FIG. 7 illustrates the primary components of a Universal Resource Locator ("URL").

FIG. 8 is a block diagram of a computer system for practicing the preferred embodiment of the present invention.

FIG. 9 is a flow diagram which illustrates the preferred steps taken to access a portion of a destination document identified in a remotely specified named anchor, even when the destination document does not contain a "NAME" attribute.

## DETAILED DESCRIPTION

### Overview Of The Preferred Method

Embodiments of the present invention use a new extension to the HTML language to support remotely specified named anchors. A remotely specified named anchor, when embedded within a source document, instructs a browser program to access a portion of a destination document indicated in the remotely specified named anchor. When the browser program reads a remotely specified named anchor such as:

```
<a href=http://foo.com/bar.html SCROLL="Some Text">
```

from the source document, the browser program performs the following steps: 1) the browser retrieves the destination file "bar.html" from the server "foo.com", 2) the browser searches the file bar.html for "Some Text", and 3) if the browser finds the character string being searched for, then the browser displays the file bar.html, scrolled to the line containing the first character of the character string being searched for.

One benefit of the present invention over previously implemented named anchors is that embodiments of the present invention provide this functionality even when the indicated portion of the destination document does not contain a "NAME" attribute. In this way, an author of a source document can create a hot link which scrolls to an indicated portion of a destination document even though the author of the source document is unable to modify, or have modified, the source code of the destination document to include a "NAME" attribute.

### Overview Of The Preferred System

FIG. 8 is a block diagram of a computer system **800** for practicing the preferred embodiment of the present invention. The computer system **800** includes a user computer **801**, a source document server computer **803**, a destination document server computer **805**, and a network communications mechanism **807**.

The user computer **801** includes a processor **809**, a memory **811**, and an interface **813** for facilitating input and output in the user computer **801**. The memory **811** stores a number of items, including a browser **815**, and an operating system **817**. The preferred browser is a Java™ enabled browser such as Hot Java™ from Sun Microsystems, Inc., of Mountain View, Calif.[1] The preferred operating system is the Solaris™ operating system from Sun Microsystems, Inc.

1. Sun and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc., in the United States and other countries.

The source document computer **803** includes a processor **819**, a memory **821**, and an interface **823** for facilitating input and output in the source computer **803**. The memory **821** stores a number of items, including a source document **825**, and an operating system **827**. The preferred operating system is the Solaris™ operating system from Sun Microsystems, Inc. of Mountain View, Calif.

The preferred source document is a text document interspersed with constructs of the HTML markup language. Another possibility would be a text document marked up with SGML (Standard Generalized Markup Language). In general, this embodiment does not require that the source document is encoded in HTML, it is preferred, however, that the document contain one or more URLs. For example, this patent application could be a source document, and in fact it would be quite convenient to be able to refer in this patent application to specific examples of web pages, hot links, and reference anchors. If the source document is not encoded in HTML, SGML, or some other standard format, it becomes more difficult, but certainly not impossible, to recognize the URLs.

This embodiment of the invention does rely on the information being represented as text, though there is no requirement that the text be encoded in ASCII. For use with other languages, text may be encoded in Unicode (the preferred embodiment for non-European languages) or any other text encoding scheme that has the simple property of allowing the computer to compare a string with a substring of the entire file and determine whether the string is identical to the substring.

The destination document computer **805** includes a processor **829**, a memory **831**, and an interface **833** for facilitating input and output in the destination computer **805**. The memory **831** stores a number of items, including a destination document **835**, and an operating system **837**. The preferred destination document is a text document interspersed with constructs of the HTML markup language. The preferred operating system is the Solaris™ operating system from Sun Microsystems, Inc. of Mountain View, Calif. The network communications mechanism **807** provides a mechanism for facilitating communication between the user computer **801**, the source document server **803**, and the destination document server **805**.

It should be noted that the user computer **801**, the source document server **803**, and the destination document server **805** may all contain additional components not shown in FIG. 8. For example, each computer could also include some combination of additional components including a video display device, an input device, such as a keyboard, mouse, or pointing device, a CD-ROM drive, and a permanent storage device, such as a disk drive.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The preferred operation of the system in FIG. 8 is perhaps best described by way of example. FIG. 9 is a flow diagram which illustrates the preferred steps taken to access a portion of a destination document identified in a remotely specified named anchor, even when the destination document does not contain a "NAME" attribute. First, a browser program reads a remotely specified named anchor from a source document. Then the browser parses the remotely specified named anchor and retrieves the name of the file to access, and the network location of the server that stores the file. After retrieving the file, the browser searches the file for the indicated text. If the indicated text is found, then the browser displays the file starting at the first character of the indicated text.

In step **901** the browser retrieves a source document. Typically, the source document will be identified by a URL supplied by the user in an "Open File" dialog box displayed by the browser. In step **903** the browser displays the source document on the user's computer. In step **905** the browser receives input by the user on the displayed source document. In step **907** the browser determines whether the user selected a hot link containing a remotely specified named anchor. If the user requested an operation other than selecting a hot link containing a remotely specified anchor then the browser merely performs the requested operation using techniques available in the prior art (step **909**). If, however, the user did select a hot link containing a remotely specified named

anchor then, in steps 911 through 923, the browser processes the remotely specified named anchor in order to access a specified portion of a destination document identified in the remotely specified named anchor, even though the specified portion of the destination document does not contain an HTML "NAME" attribute associated with it.

In step 911 the browser parses the remotely specified named anchor to obtain the name of the file to retrieve, as well as the name of the server storing the file. In step 913 the browser retrieves the file from the server. In step 915 the browser parses the remotely specified named anchor in order to retrieve the character string for which to search. Those of ordinary skill in this art will understand that, alternatively, the browser could, in step 911, parse the remotely specified named anchor to retrieve the character string to search for. In step 917 the browser searches the retrieved file for the character string. If the character string is not found in the file (step 919) then the retrieved file is displayed to the user starting at the top of the file (step 921). If, however, the character string is found in the retrieved file then the browser displays the file scrolled to the line containing the first occurrence of the character string being searched for (step 923).

In this way, a new method and system are provided which access a portion of a destination document identified in a remotely specified named anchor, even when the destination document does not contain a "NAME" attribute.

One weakness of the preferred embodiment is that it does not allow the author of the source document to point to a second occurrence of a character string. Consider as an example the following file:

1. Socrates was a man.
2. All men are mortal.
3. Therefore, Socrates was mortal.
4. So his ultimate downfall was due to the fact that Socrates was a man. For example, the preferred embodiment will not link to the character string "Socrates was a man" in line 4 in response to the remotely specified named anchor

```
<a href=http://foo.com/socratestory.html SCROLL:"Socrates was a man">
```

because the preferred embodiment will instead scroll to the first occurrence of the string "Socrates was a man" in line 1 of the file. This limitation is not severe, however, since it will normally be the case that the author can merely keep adding to the character string of choice until it is uniquely identified. For example, if it indeed was desired to link to the occurrence of "Socrates was a man" in line 4, then the author could merely search for the string "that Socrates was a man". In this way, the preferred embodiment would scroll the file to line 4, as desired. Though not a perfect solution, this solution will be adequate in almost all cases.

In general, embodiments of the invention apply to any system where it is desired to be able to point to a specific part of a larger whole even when one cannot get access to this larger whole to insert a reference marker.

There is an alternative, much simpler, way of solving one aspect addressed by the present invention, but it is *not* recommended. One could simply point to the character position (offset) of the desired scroll within the destination file. The reason this is not recommended is that the character position will change if the owner of the destination file edits it. Editing the file may also change the search string but this is much less likely to happen than the more common case where the author adds or deletes text in another part of the document.

While specific embodiments have been described herein for purposes of illustration, various modifications may be made without departing from the spirit and scope of the invention. Accordingly, the invention is not limited to the above described embodiments, but instead is defined by the claims which follow, along with their full scope of equivalents.

What is claimed is:

1. A method executed in a network computer system for facilitating access to a specified portion of data stored at a remote location, the method comprising the steps of:
   - retrieving a source document, the source document including hypertext links to other data on the network;
   - displaying the source document;
   - receiving input entered on the source document;
   - determining whether the input comprises selection of a remotely specified named anchor;
   - when the input comprises selection of a remotely specified named anchor, retrieving data indicated in the remotely specified named anchor and displaying a portion of the data specified in the remotely specified named anchor, wherein the specified portion of the data does not have a position marker associated with it.

2. The method of claim 1 wherein the step of determining further comprises the step of:
   - examining the remotely specified named anchor to determine whether it contains an attribute indicating that a specified portion of the retrieved data should be displayed.

3. The method of claim 2 wherein the attribute is a SCROLL attribute.

4. The method of claim 1 wherein the step of displaying the portion of the data specified in the remotely specified named anchor further comprises the step of:
   - examining the remotely specified named anchor to determine a character string to search for.

5. The method of claim 4 further comprising the steps of:
   - searching the retrieved data for the character string; and
   - displaying the portion of the data containing the character string.

6. A network computer system for facilitating access to a specified portion of data stored at a remote location, the system comprising:
   - a mechanism configured to retrieve a source document, the source document including hypertext links to other data on the network;
   - a mechanism configured to display the source document;
   - a mechanism configured to receive input entered on the source document;
   - a mechanism configured to determine whether the input comprises selection of a remotely specified named anchor;
   - a mechanism configured to, when the input comprises selection of a remotely specified named anchor, retrieve data indicated in the remotely specified named anchor and display a portion of the data specified in the remotely specified named anchor, wherein the specified portion of the data does not have a position marker associated with it.

7. The system of claim 6 wherein the mechanism configured to determine further comprises:
   - a mechanism configured to examine the remotely specified named anchor to determine whether it contains an attribute indicating that a specified portion of the retrieved data should be displayed.

8. The system of claim 7 wherein the attribute is a SCROLL attribute.

9. The system of claim 6 wherein the mechanism configured to display the portion of the data specified in the remotely specified named anchor further comprises:

a mechanism configured to examine the remotely specified named anchor to determine a character string to search for.

10. The system of claim 9 further comprising:

a mechanism configured to search the retrieved data for the character string; and

a mechanism configured to display the portion of the data containing the character string.

11. A computer program product for facilitating access to a specified portion of data stored at a remote location, the computer program product comprising:

code that retrieves a source document, the source document including hypertext links to other data on the network;

code that displays the source document;

code that receives input entered on the source document;

code that determines whether the input comprises selection of a remotely specified named anchor;

code that, when the input comprises selection of a remotely specified named anchor, retrieves data indicated in the remotely specified named anchor and displays a portion of the data specified in the remotely specified named anchor, wherein the specified portion of the data does not have a position marker associated with it,

wherein the code resides on a tangible medium.

12. The computer program product of claim 11 wherein the code that determines further comprises:

code that examines the remotely specified named anchor to determine whether it contains an attribute indicating that a specified portion of the retrieved data should be displayed.

13. The computer program product of claim 12 wherein the attribute is a SCROLL attribute.

14. The computer program product of claim 11 wherein the code that displays the portion of the data specified in the remotely specified named anchor further comprises:

code that examines the remotely specified named anchor to determine a character string to search for.

15. The computer program product of claim 14 further comprising:

code that searches the retrieved data for the character string; and

code that displays the portion of the data containing the character string.

* * * * *

US006278993B1

(12) **United States Patent**
**Kumar et al.**

(10) **Patent No.: US 6,278,993 B1**
(45) **Date of Patent: Aug. 21, 2001**

(54) **METHOD AND APPARATUS FOR EXTENDING AN ON-LINE INTERNET SEARCH BEYOND PRE-REFERENCED SOURCES AND RETURNING DATA OVER A DATA-PACKET-NETWORK (DPN) USING PRIVATE SEARCH ENGINES AS PROXY-ENGINES**

(75) Inventors: **Srihari Kumar**; **Sreeranga P. Rajan**, both of Santa Clara, CA (US)

(73) Assignee: **Yodlee.com, Inc.**, Redwood Shores, CA (US)

( * ) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/497,089**

(22) Filed: **Feb. 3, 2000**

**Related U.S. Application Data**

(63) Continuation-in-part of application No. 09/323,598, filed on Jun. 1, 1999, which is a continuation-in-part of application No. 09/208,740, filed on Dec. 8, 1998.

(51) **Int. Cl.**[7] ..... **G06F 15/00**; G06F 17/30

(52) **U.S. Cl.** ..... **707/3**; 707/501; 707/5

(58) **Field of Search** ..... 707/1–10, 100–104, 707/500–534; 345/326–335; 709/200–217

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,793,966 * 11/1998 Amstein et al. ..... 709/203
5,832,494 * 11/1998 Egger et al. ..... 707/102
5,991,756 * 11/1999 Wu ..... 707/3

OTHER PUBLICATIONS

O'Leary, Mick, "NewsWorks, brings new depth to Web news; the site excels with unique sources and value–added editorial features", Information Today, v14, p. 10, Dec. 1997.*

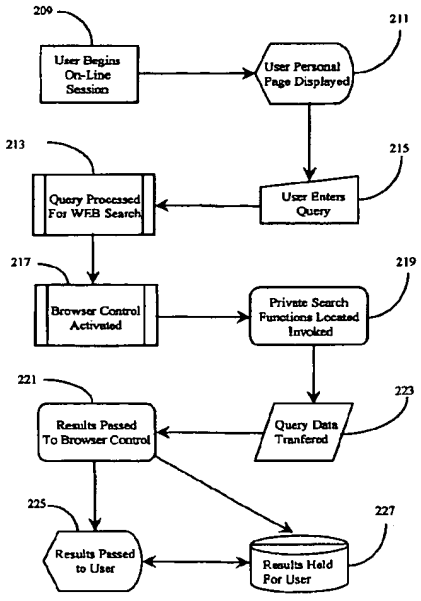
* cited by examiner

*Primary Examiner*—Hosain T. Alam
*Assistant Examiner*—Alford W. Kindred
(74) *Attorney, Agent, or Firm*—Donald R Boys; Central Coast Patent Agency, Inc.

(57) **ABSTRACT**

A search function is provided for Internet searching capable of searching to greater depth than conventional search functions. The new function tests returned electronic documents from a first search for a second search function, and, finding a second function, transfers at least a form of first search criteria into the second search function, then initiated the second function, and returns at least addresses of documents found by the second function into the first function. In a preferred embodiment a search function according to the invention is provided by a subscription portal server, and operates by proxy, initiated and controlled by subscribers. In this form, primary searches may be limited to destinations registered to specific subscribers using the function.

**11 Claims, 11 Drawing Sheets**



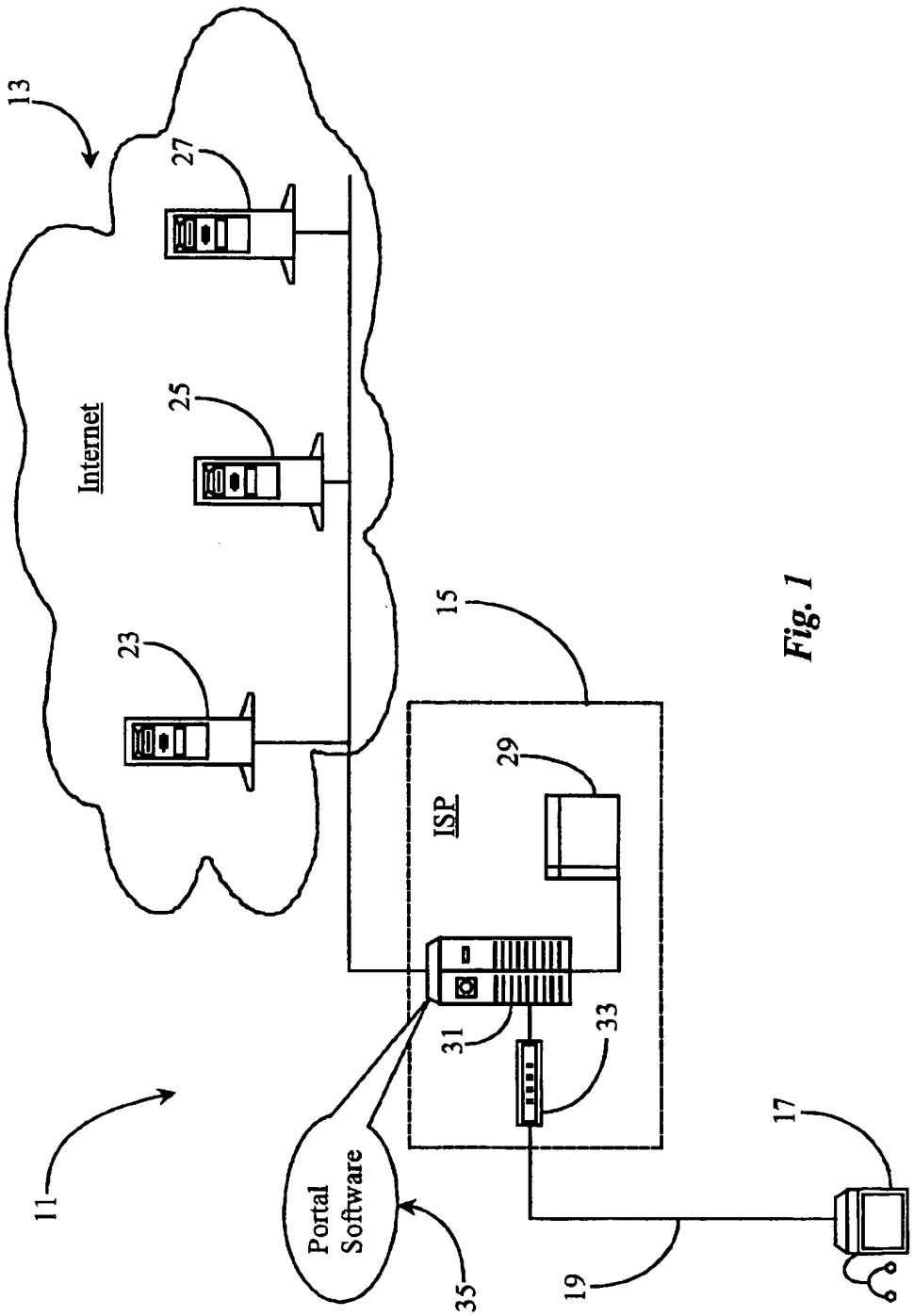



Fig. 1

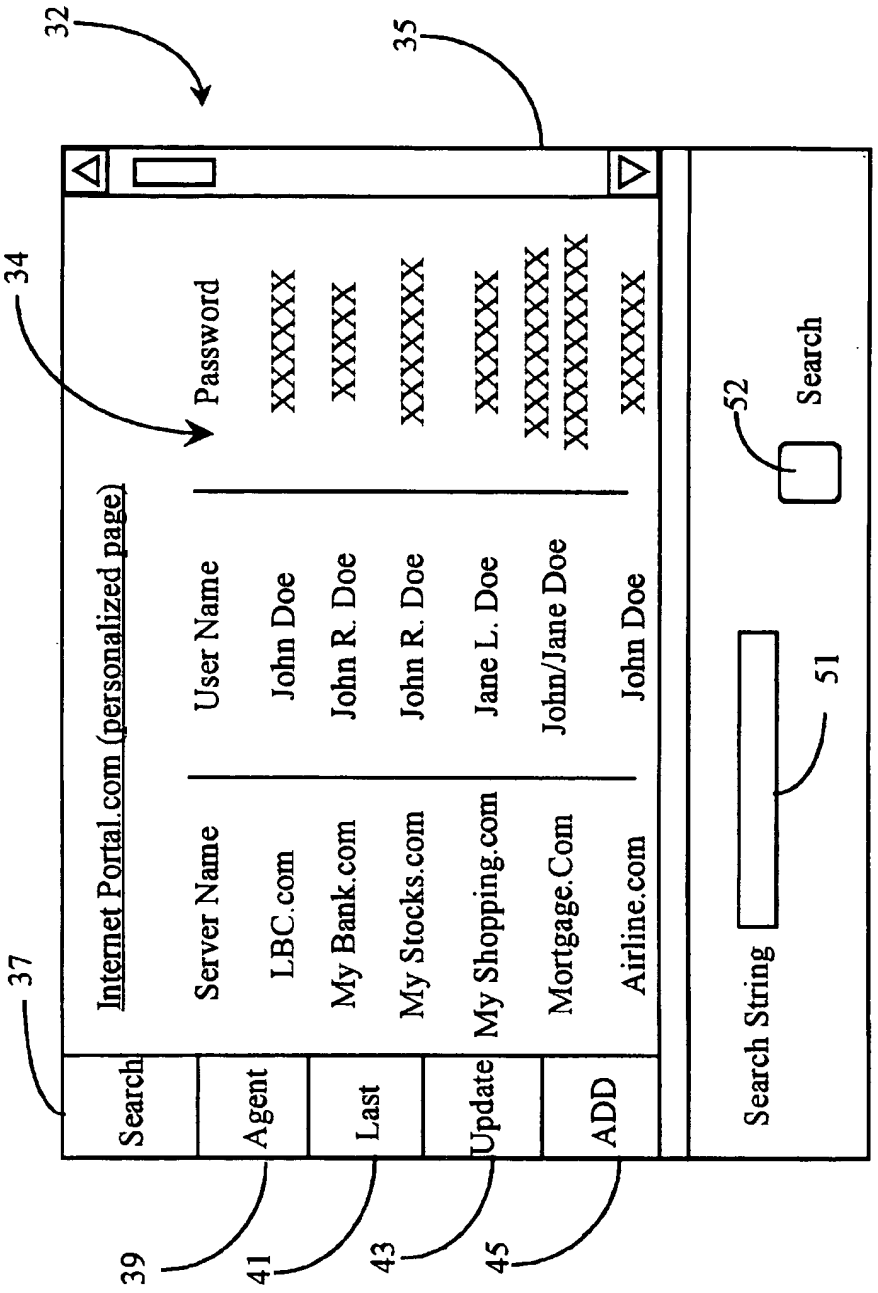

32
35
34
37
Internet Portal.com (personalized page)
Server Name
User Name
Password
LBC.com
John Doe
XXXXXX
My Bank.com
John R. Doe
XXXX
My Stocks.com
John R. Doe
XXXXXXX
My Shopping.com
Jane L. Doe
XXXXXX
Mortgage.Com
John/Jane Doe
XXXXXXXX
Airline.com
John Doe
XXXXXXXXX
XXXXXX
Search
Agent
Last
Update
ADD
39
41
43
45
Search String
51
52
Search

*Fig. 2*

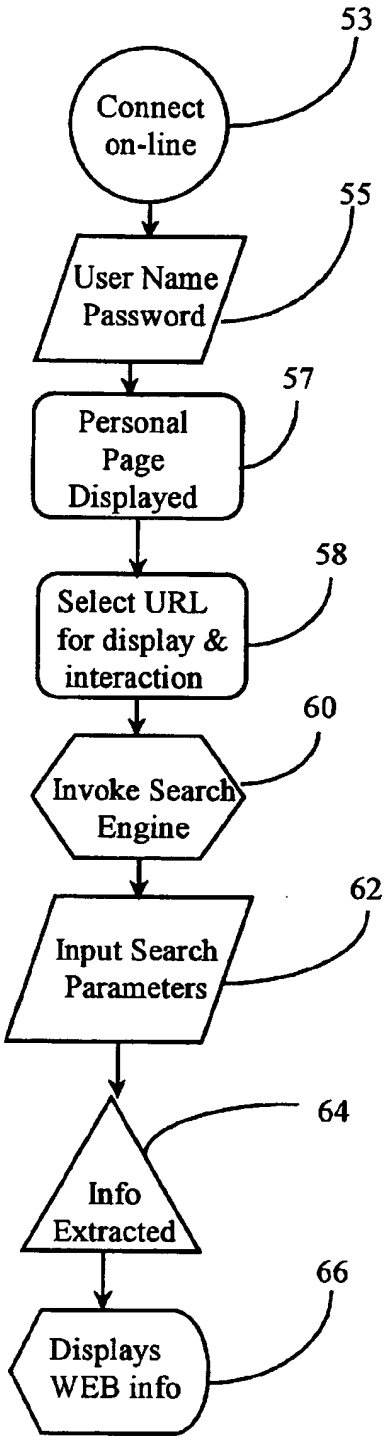

53
Connect on-line
55
User Name Password
57
Personal Page Displayed
58
Select URL for display & interaction
60
Invoke Search Engine
62
Input Search Parameters
64
Info Extracted
66
Displays WEB info

*Fig. 3*

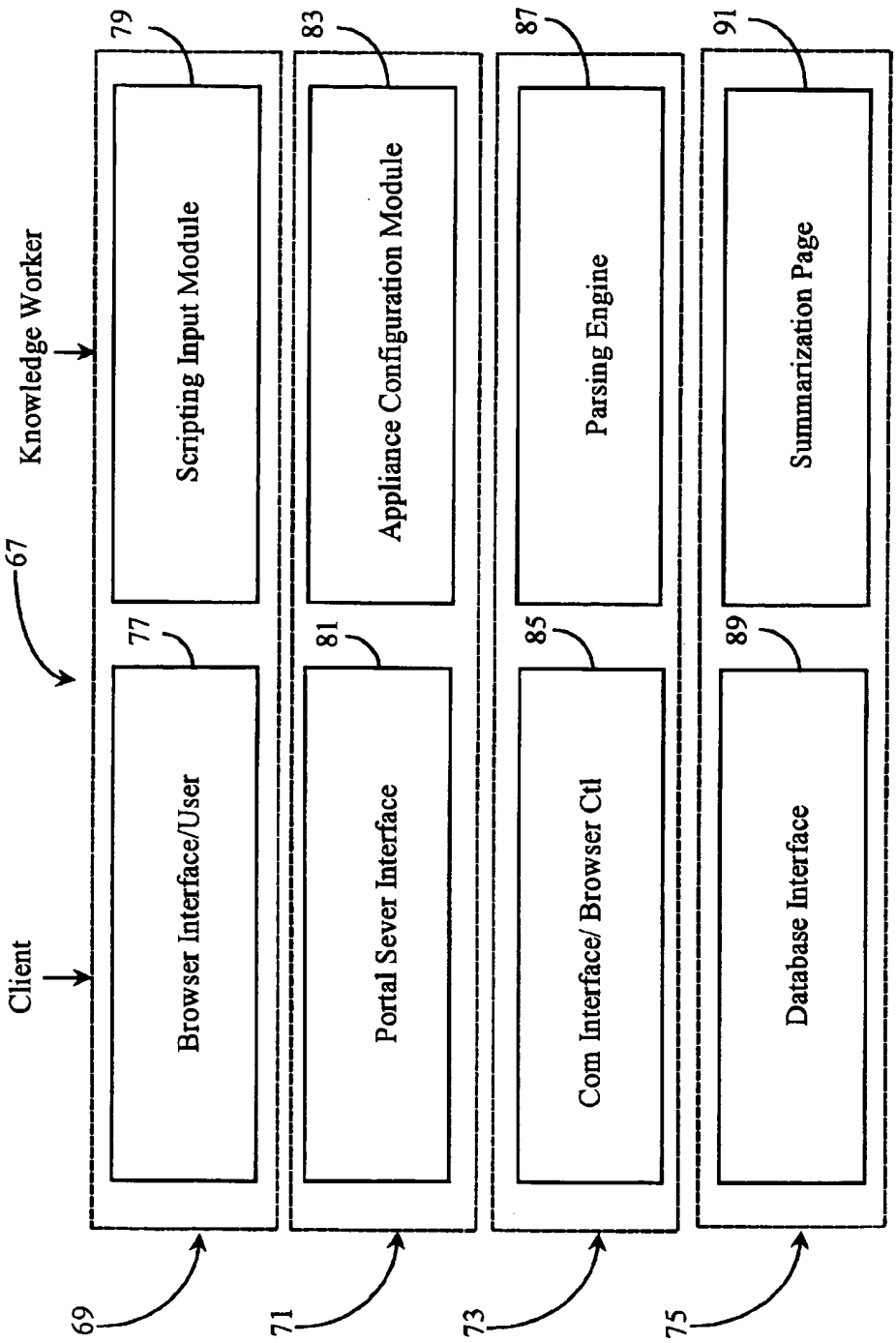

Knowledge Worker
Client
67
79
Scripting Input Module
83
Appliance Configuration Module
87
Parsing Engine
91
Summarization Page
77
Browser Interface/User
81
Portal Sever Interface
85
Com Interface/ Browser Ctl
89
Database Interface
69
71
73
75

*Fig. 4*

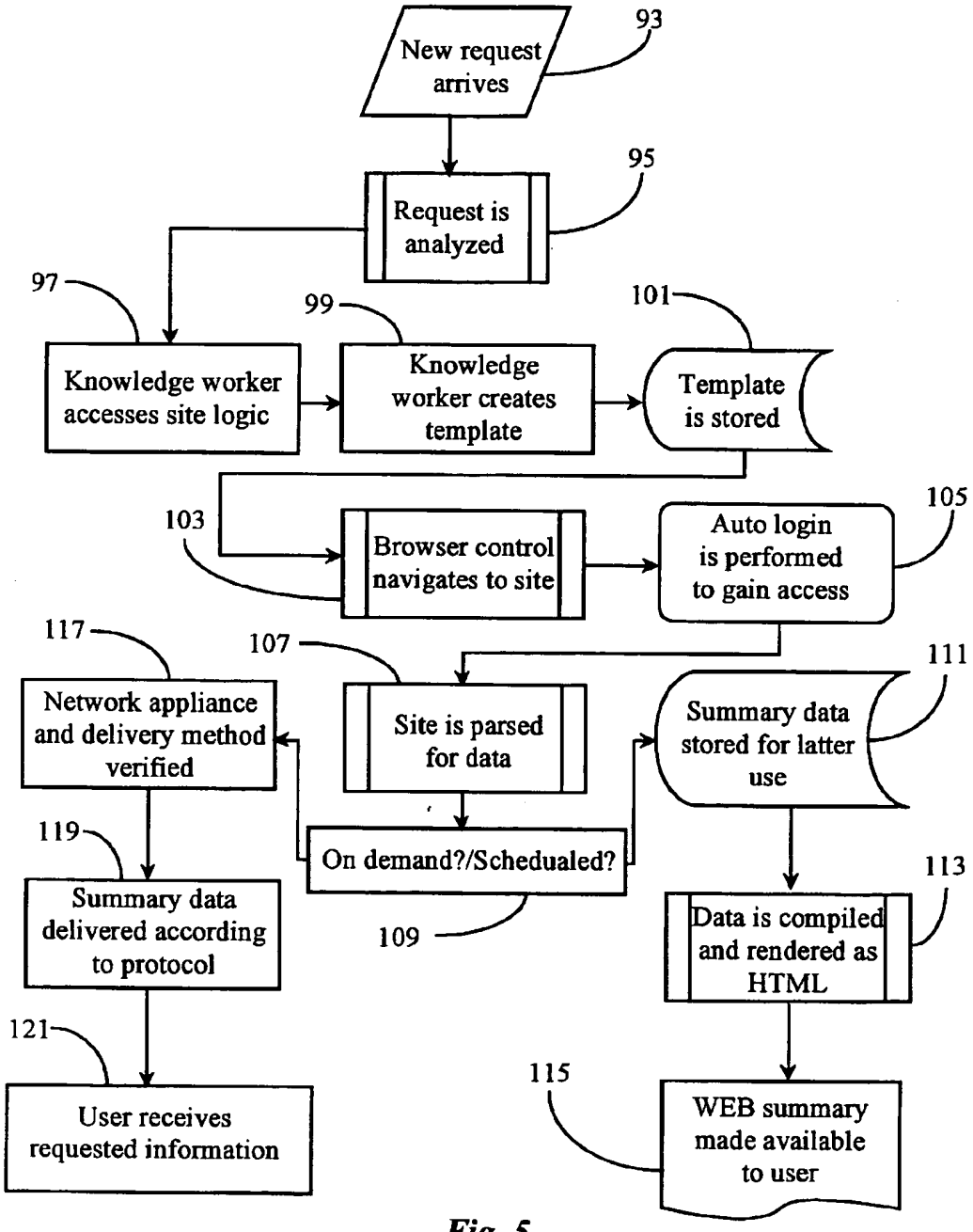

New request arrives
93
Request is analyzed
95
97
Knowledge worker accesses site logic
99
Knowledge worker creates template
101
Template is stored
103
Browser control navigates to site
Auto login is performed to gain access
105
117
Network appliance and delivery method verified
107
Site is parsed for data
111
Summary data stored for latter use
119
Summary data delivered according to protocol
On demand?/Schedualed?
109
113
Data is compiled and rendered as HTML
121
User receives requested information
115
WEB summary made available to user

*Fig. 5*

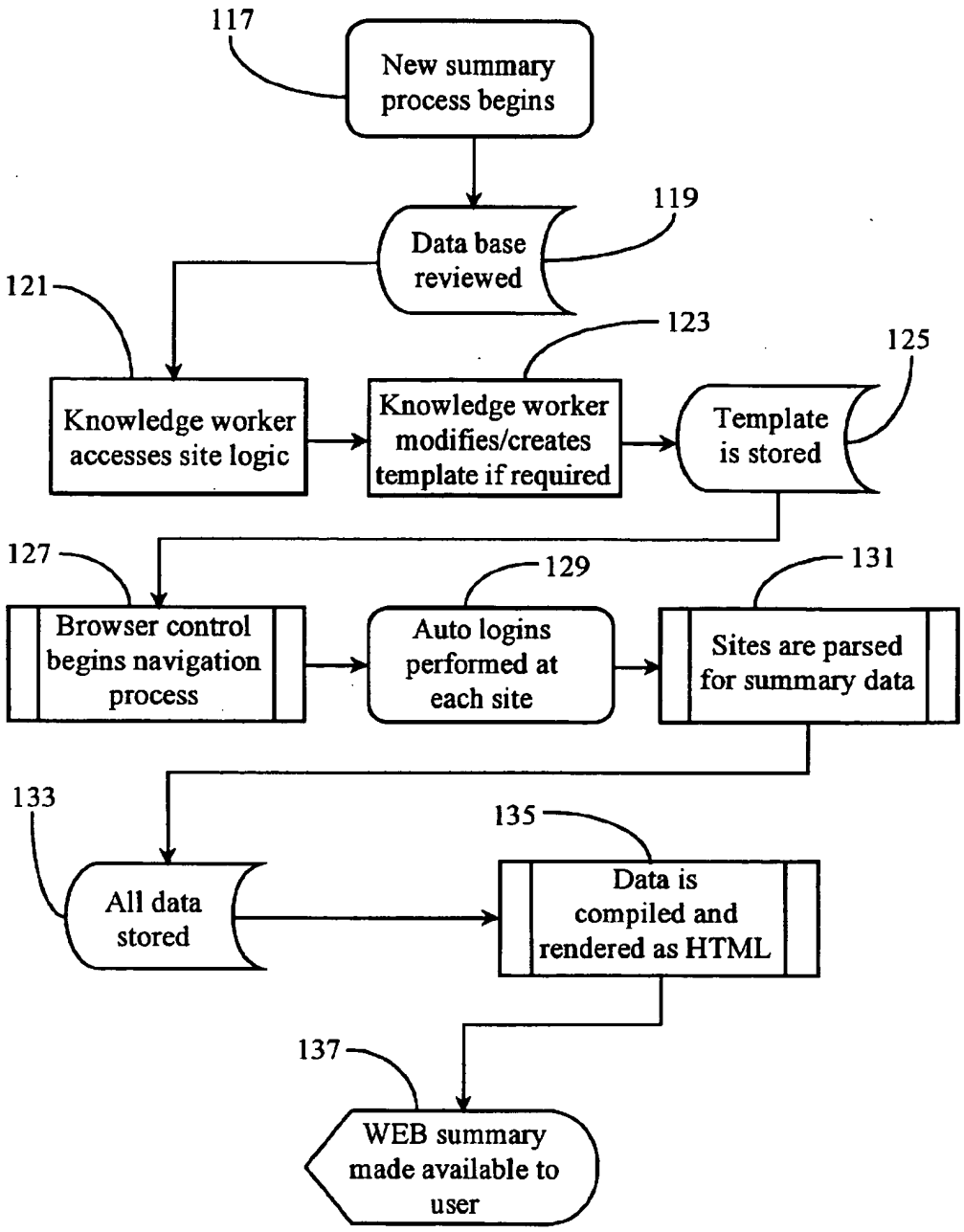

117
New summary process begins
119
Data base reviewed
121
123
125
Knowledge worker accesses site logic
Knowledge worker modifies/creates template if required
Template is stored
127
129
131
Browser control begins navigation process
Auto logins performed at each site
Sites are parsed for summary data
133
135
All data stored
Data is compiled and rendered as HTML
137
WEB summary made available to user

*Fig. 6*

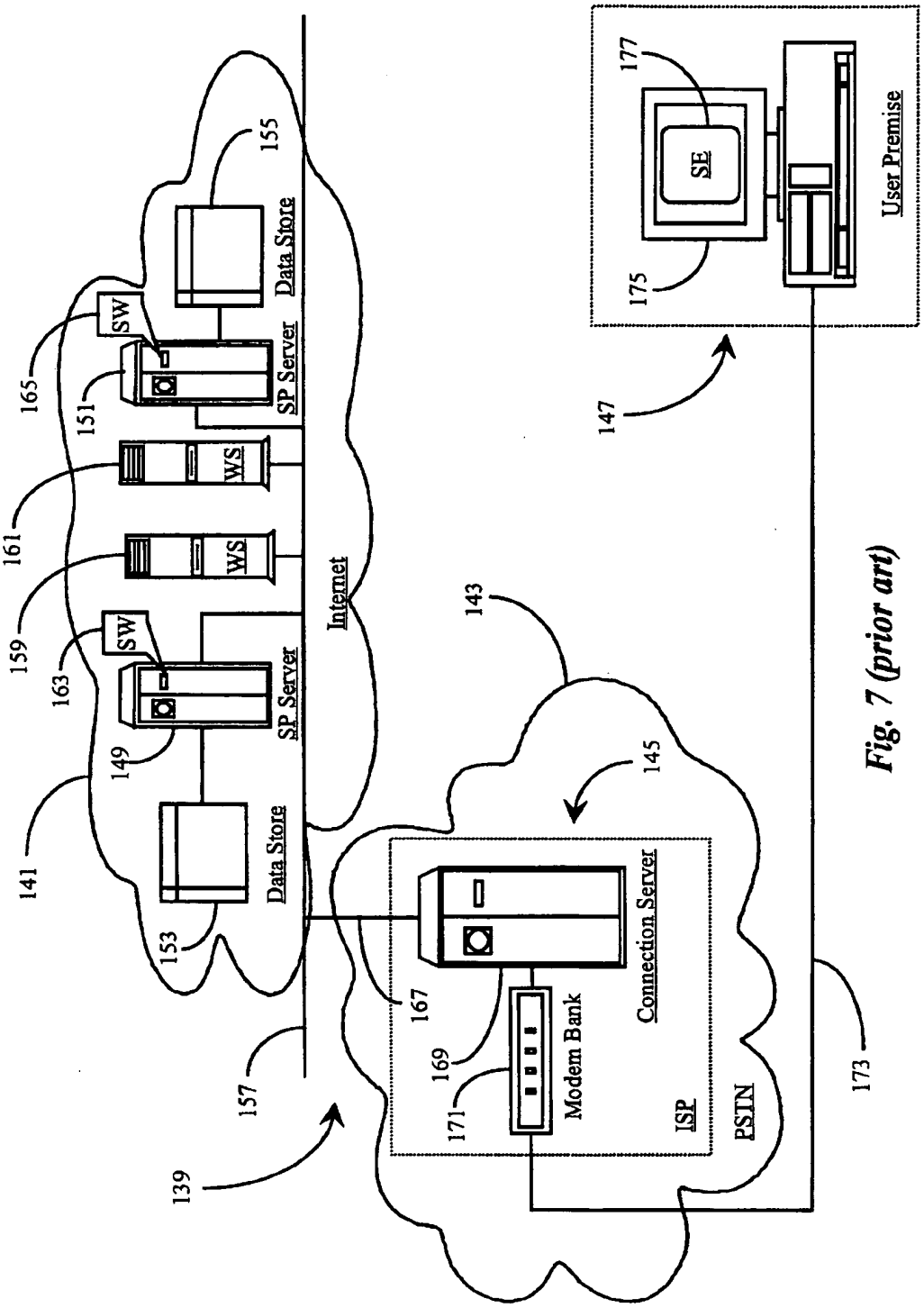



*Fig. 7 (prior art)*

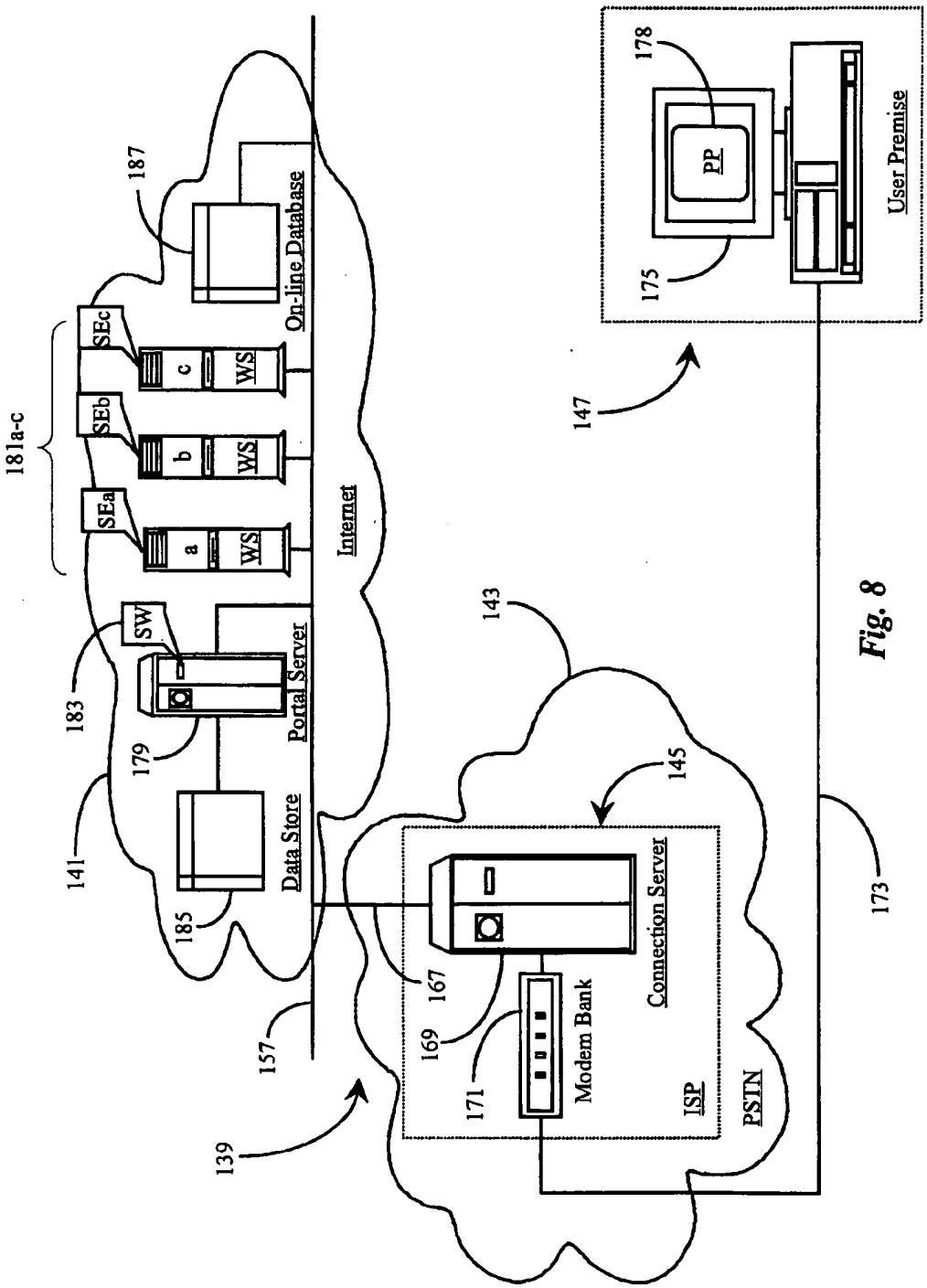

181a-c
SEc
SEb
SEa
c
b
a
WS
WS
WS
187
On-line Database
Internet
SW
183
179
Portal Server
141
185
Data Store
157
139
143
145
Connection Server
167
169
171
Modem Bank
ISP
PSTN
173
147
178
PP
175
User Premise

*Fig. 8*

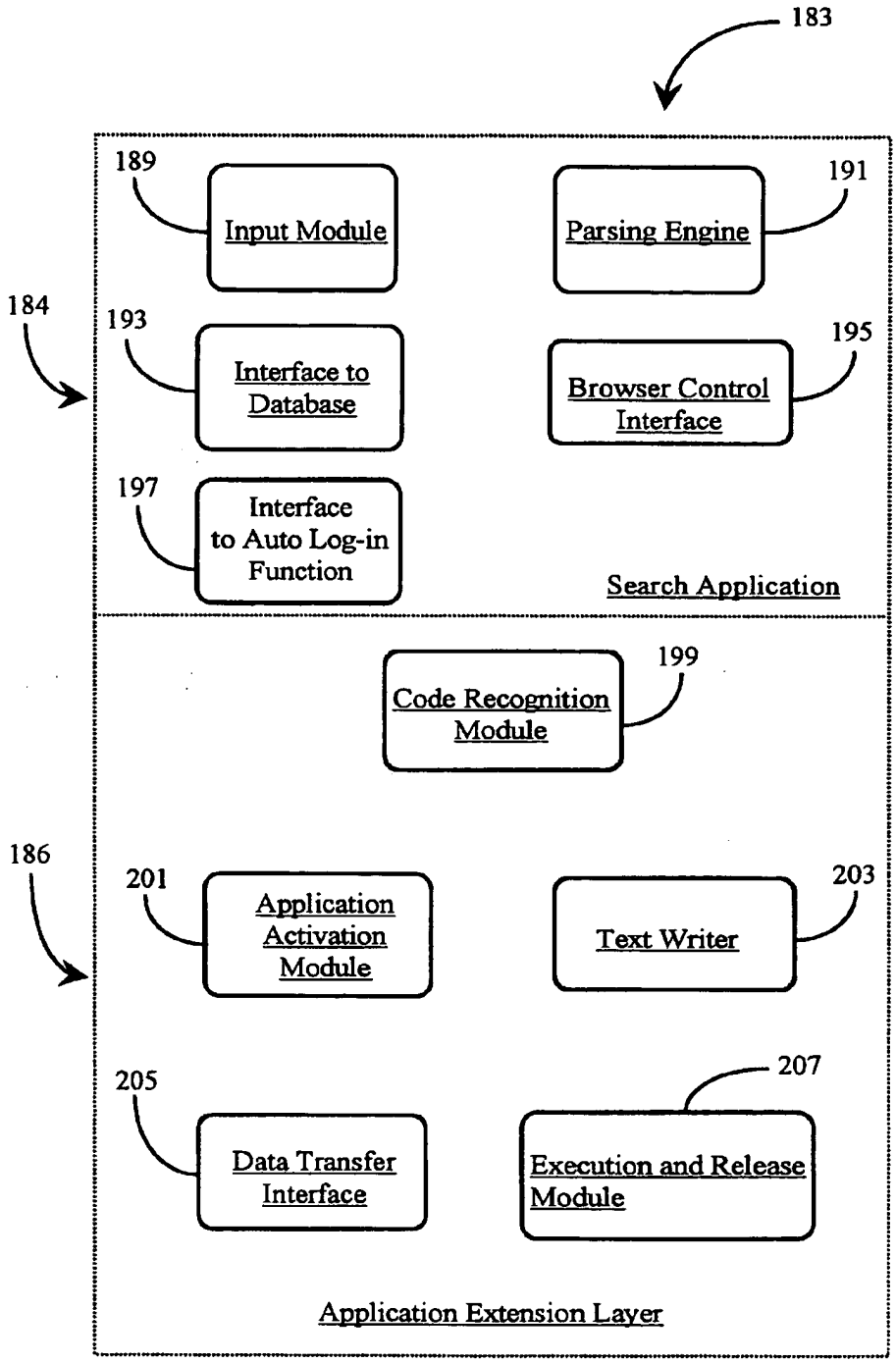

183
189
Input Module
191
Parsing Engine
184
193
Interface to Database
195
Browser Control Interface
197
Interface to Auto Log-in Function
Search Application
199
Code Recognition Module
186
201
Application Activation Module
203
Text Writer
205
Data Transfer Interface
207
Execution and Release Module
Application Extension Layer

*Fig. 9*

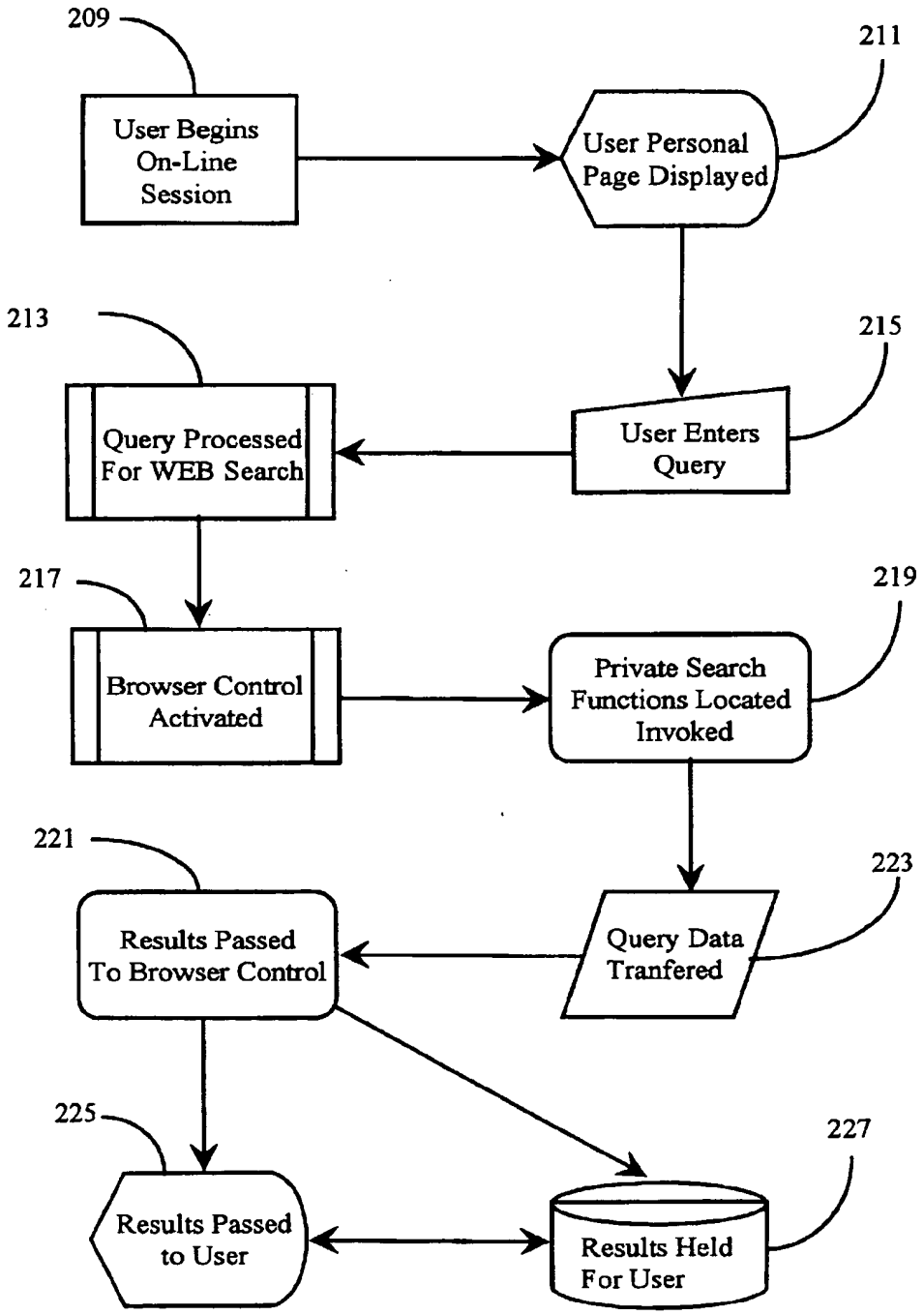

209
User Begins On-Line Session
211
User Personal Page Displayed
213
Query Processed For WEB Search
215
User Enters Query
217
Browser Control Activated
219
Private Search Functions Located Invoked
221
Results Passed To Browser Control
223
Query Data Tranfered
225
Results Passed to User
227
Results Held For User

*Fig. 10*

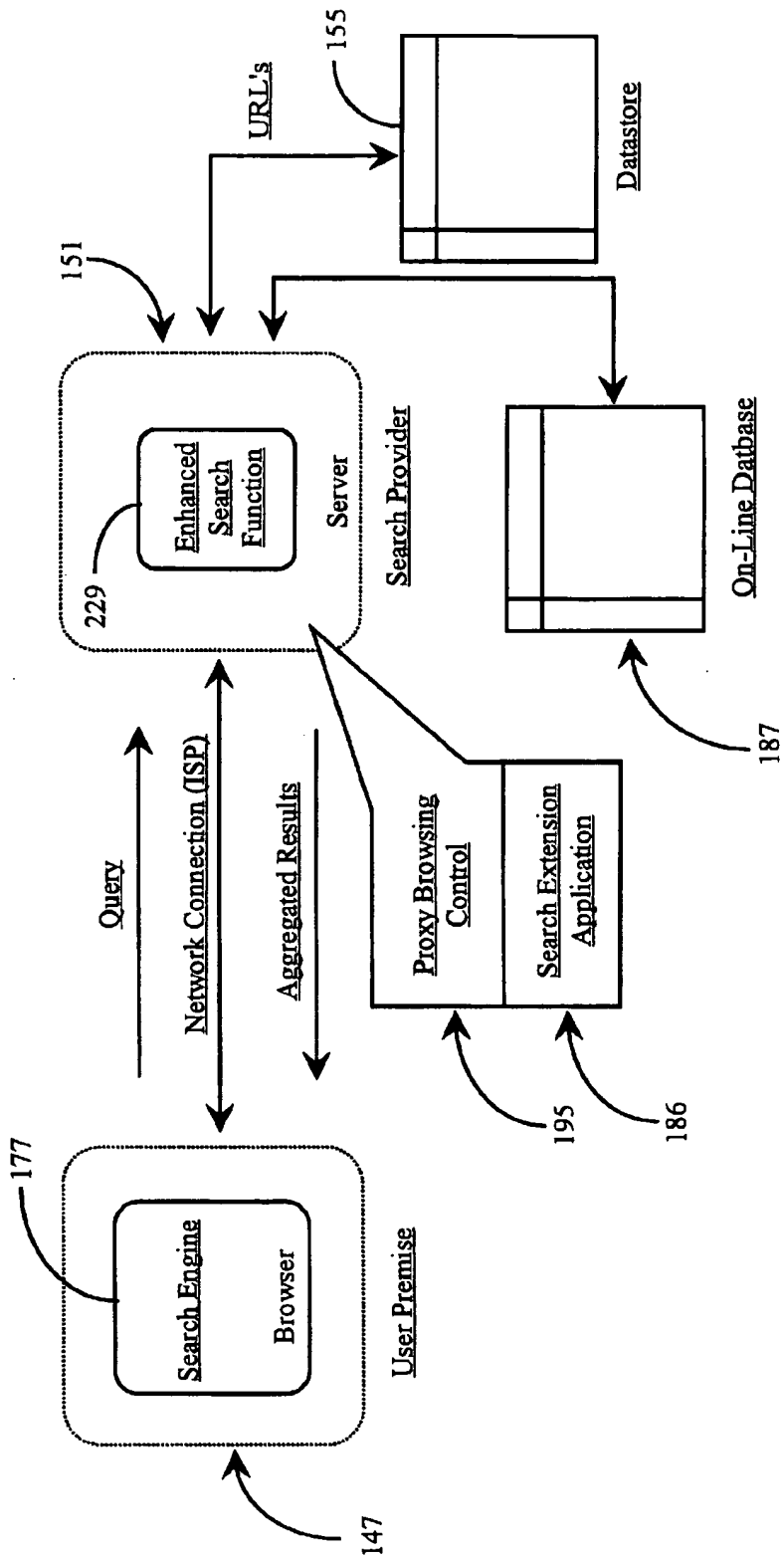

155
URL's
Datastore
151
Enhanced Search Function
229
Server
Search Provider
On-Line Database
187
Query
Network Connection (ISP)
Aggregated Results
Proxy Browsing Control
Search Extension Application
195
186
177
Search Engine
Browser
User Premise
147

*Fig. 11*

# METHOD AND APPARATUS FOR EXTENDING AN ON-LINE INTERNET SEARCH BEYOND PRE-REFERENCED SOURCES AND RETURNING DATA OVER A DATA-PACKET-NETWORK (DPN) USING PRIVATE SEARCH ENGINES AS PROXY-ENGINES

## CROSS-REFERENCE TO RELATED DOCUMENTS

The present invention is a continuation in part (CIP) to patent application Ser. No. 09/323,598 entitled "Method and Apparatus for Obtaining and Presenting WEB Summaries to Users" filed on Jun. 1, 1999, which is a CIP to patent application Ser. No. 09/208,740 entitled "Method and Apparatus for Providing and Maintaining a User-Interactive Portal System Accessible via Internet or other Switched-Packet-Network" filed on Dec. 8, 1998, disclosures of which are incorporated herein in their entirety by reference.

## FIELD OF THE INVENTION

The present invention is in the field of Internet navigation and data gathering over a DPN network and pertains more particularly to a method and apparatus for searching for and returning data associated with URL's not indexed in traditional search engine databases, by using secondary proxy engines.

## BACKGROUND OF THE INVENTION

The information network known as the World Wide Web (WWW), which is a subset of the well-known Internet, is arguably the most complete source of publicly accessible information available. Anyone with a suitable Internet appliance such as a personal computer with a standard Internet connection may access (go on-line) and navigate to Universal Resource Locators (URL's), also termed information pages or WEB pages, stored on Internet-connected servers for the purpose of garnering information and initiating transactions with hosts of such servers and pages.

Many companies offer various subscription services accessible via the Internet. For example, many people now do their banking, stock trading, shopping, and so forth from the comfort of their own homes via Internet access. Typically, a user, through subscription, has access to personalized and secure WEB pages for such functions. By typing in a user name and a password or other personal identification code, a user may obtain information, initiate transactions, buy stock, and accomplish a myriad of other tasks.

One problem that is encountered by an individual who has several or many such subscriptions to Internet-brokered services is that there are invariably many passwords and/or log-in codes to be used. Often a same password or code cannot be used for every service, as the password or code may already be taken by another user. A user may not wish to supply a code unique to the user such as perhaps a social security number because of security issues, including quality of security, that may vary from service to service. Additionally, many users at their own volition may choose different passwords for different sites so as to have increased security, which in fact also increases the number of passwords a user may have.

Another issue that can plague a user who has many passworded subscriptions is the fact that they must bookmark many WEB pages in a computer cache so that they



may quickly find and access the various services. For example, in order to reserve and pay for airline travel, a user must connect to the Internet, go to his/her book-marks file and select an airline page. The user then has to enter a user name and password, and follow on-screen instructions once the page is delivered. If the user wishes to purchase tickets from the WEB site, and wishes to transfer funds from an on-line banking service, the user must also look for and select the personal bank or account page to initiate a funds transfer for the tickets. Different user names and passwords may be required to access these other pages, and things get quite complicated.

Although this preceding example is merely exemplary, it is generally known that much work related to finding WEB pages, logging in with passwords, and the like is required to successfully do business on the WEB.

A service known to the inventor and described in the related case listed under the cross-reference to related documents section provides a WEB service that allows a user to store all of his password protected pages in one location such that browsing and garnering information from them is much simplified. A feature of the above service allows a user to program certain tasks into the system such that requested tasks are executed by an agent (software) based on user instruction. The service stores user password and log-in information and uses the information to log-in to the user's sites, thus enabling the user to navigate without having to manually input log-in or password codes to gain access to the links.

The above-described service uses a server to present a user-personalized application that may be displayed as an interactive home page that contains all of his listed sites (hyperlinks) for easy navigation. The application lists the user's URL's in the form of hyperlinks such that a user may click on a hyperlink and navigate to the page wherein log-in, if required, is automatic, and transparent to the user.

The application described above also includes a software agent that may be programmed to perform scheduled tasks for the user including returning specific summaries and updates about user-account pages. A search function is provided and adapted to cooperate with the software agent to search user-entered URL's for specific content if such pages are cached somewhere in their presentable form such as at the portal server, or on the client's machine.

An enhancement to the personalized system described above allows a software agent termed a gatherer agent (browser navigation control) to, in cooperation with a search function, navigate by proxy to any user-entered URL and return updated data back to the user in the form of an HTML information pace, which appears in the user's browser window. The enhancement is accomplished with the use of site-logic scripting based on pre-known information about the URL or URL's from which a user wishes to obtain data. In this way, current data specifically requested by a user may be found and retrieved for the user.

The process described above is initiated by a user query that is entered into a search function dialog box provided with a user's personal portal page. The query may be presented in natural language adding a level of user friendliness to the process. Moreover, auto log-in to password protected sites may be performed on behalf of users by virtue of the system's compilation and storage of user and WEB-site related data.

A limitation exists in the personalized system described above in that the search function may not search beyond the indexed or known URLs listed in the service database and

attributed to a requesting user. That is to say that the search function cannot proceed beyond the first level of WEB site depth. Therefore, manual navigation must still be performed by a user who desires to obtain data referenced at a deeper level than in an indexed or registered URL. Of course, a user practicing the above system may physically register any new URLs with the service such that they may be included in the search criteria and site logic may be developed for obtaining summary data contained in the new URLs. However, when performing a general search, a user may not know the URL where the desired data is held.

In a general sense, as opposed to the personalized system described above, the current technology of searching URLs over a DPN such as the Internet with a search engine involves entering a query into a search dialog box and submitting the query to a server hosted by the provider of the search function. The requested data is compared to data held in a database containing cached URLs, which may contain data matching a query. Matching URLs (URLs with data content matching a query) are then returned to a user's browser window for browsing and selection as is generally known in the art.

There are many different methods and criteria used by search engines for searching out data on the Internet. Most typical is the use of key words or phrases that are used to find matches in text contained on an indexed WEB page (URL). Other methods include searching by site, searching for video, searching for photographs, searching for audio, and so on.

The above technology is limited in a general sense as described above by a fact that all URL pages containing information which may be desired by a user, are not listed in conventional search engine databases. In fact, there are a vast number of URL resources that are maintained on the Internet that are not listed in any search engine database and therefore may not be found through a query-type search method.

For example, a main page having a URL and hosted by an enterprise may contain several links to pages that contain additional information.

However, only the main page of the site is typically indexed on any given search-engine list unless a host of the site or other entity submits the additional URLs to be included in a database held by the enterprise hosting the search engine. Therefore, in order to obtain the additional data from un-indexed sites, a user must navigate to additional sites from a "jump-off page" found during the original search.

Many enterprises, especially companies hosting many pages, provide a convenient search engine function embedded into a page at a main URL, which is indexed in the conventional search-engine databases. In this way, a user may search for the main site, invoke the returned link, and then use the provided private search engine to explore the additional pages or look for additional data related to the site as a whole. Such a WEB site may be a company or enterprise site comprising many related WEB pages.

When a user invokes a private search function on a main page, he must enter a new query into the private search engine to look for the additional data. He or she is no longer using the original search engine to look for the data. Moreover, the private search engine provided at a site's main URL may function by different rules then the original search engine requiring a user, in many cases, to restructure the original query.

In the personalized system described further above wherein pre-knowledge exists about the user and WEB page site-logic is known pertaining to how data is hosted at the service-site marked by a known URL, the limitation described in a general data-search system still exists. That is, URLs may not be found if they are not pre-known or indexed.

What is clearly needed is a method and apparatus that enables a search engine to find and obtain data from URLs that are not indexed by a search engine database or otherwise pre-known to a requesting user or search-hosting service. A method and apparatus such as this would allow a user to obtain data that would otherwise have to be obtained by further browser navigation.

## SUMMARY OF THE INVENTION

In a preferred embodiment of the present invention a method for extending an on-line Internet search beyond pre-referenced sources is provided, comprising steps of (a) entering a first search criteria in a first search function; (b) initiating the first search function; (c) returning in the first search function a pre-referenced first document having data associated with the first search criteria; (d) testing the first document for an embedded second search function; (e) on finding a second search function in the first document, automatically entering at least a form of the first search criteria in the second search function; and (f) returning addresses in the first search function for documents found through the second search function.

In one embodiment the first search function allows natural language in entering search criteria, and further comprises a parsing step for parsing criteria input for significant words and phrases for criteria matches. The first function, in some embodiments in step (e), tests the second search function for criteria rules, and amends the first search criteria to conform to the criteria rules.

In some embodiments the first search function is provided by a subscription portal service, an is operated by proxy by subscribers. In some of these embodiments the first search function is limited in step (c) to returning first documents pre-registered to a specific subscriber invoking the first search function.

In another aspect of the invention an Internet search application is provided, comprising a first search module having a first criteria interface for entry of a first search criteria; an inspection function for identifying a second search module in a returned electronic document, the second search module having a second criteria interface; and an entry module for entering the first search criteria into the search criteria interface of the second search module. The new search function is characterized in that the search application, upon entry of a first search criteria in the first criteria interface, returns at least one electronic document having a match to the first search criteria, inspects the document for the second search module, and transfers at least a form of the first search criteria into the second criteria interface.

In preferred embodiments the Internet search application further initiates the second search module after transfer of search criteria, and returns at least addresses of documents found by the second search function in the first search function.

In some cases the first search module allows natural language criteria entry, and parses entries for significant words and phrases for matching to content in electronic documents returned. In some cases as well, the first search module, in step (e), tests the second search module for criteria rules, and amends the first search criteria to conform to the criteria rules.

In some cases the first search module is provided by a subscription portal service, an is operated by proxy by subscribers, and in some of these embodiments the first search module is limited to returning first documents pre-registered to a specific subscriber invoking the first search function.

In embodiments of the present invention taught in enabling detail below, for the first time, a search function is provided for Internet browsing that is capable of invoking secondary and private search functions in documents returned by a first search operation, and finding documents at further depth through the invoked functions.

## BRIEF DESCRIPTION OF THE DRAWING FIGURES

FIG. **1** is an overview of an Internet portal system and network according to an embodiment of the present invention.

FIG. **2** is an exemplary plan view of a personalized Portal home page application as it may be seen on a display monitor according to an embodiment of the present invention.

FIG. **3** is a flow diagram illustrating user interaction with the Internet portal of FIG. **1**.

FIG. **4** is a block diagram illustrating a summarization software agent and capabilities thereof according to an embodiment of the present invention.

FIG. **5** is a logical flow chart illustrating an exemplary summarization process performed by the software agent of FIG. **4** operating in a user-defined mode.

FIG. **6** is a logical flow chart illustrating an exemplary summarization process performed by the software agent of FIG. **4** in a User-independent smart mode with minimum user input.

FIG. **7** is an architectural overview of a system navigated to search for data on a DPN network according to prior art.

FIG. **8** is an architectural overview of a system employing a personalized search method for data on a DPN network according to an embodiment of the present invention.

FIG. **9** is a block diagram illustrating software components of a search function interface according to an embodiment of the present invention.

FIG. **10** is a process flow diagram illustrating basic interaction steps for practicing the present invention according to a preferred embodiment.

FIG. **11** is a block diagram illustrating the standard data-search system of FIG. **7** enhanced with the method and apparatus of the present invention according to an alternate embodiment of the present invention.

## DESCRIPTION OF THE PREFERRED EMBODIMENTS

According to a preferred embodiment of the present invention, a unique Internet portal is provided and adapted to provide unique services to users who have obtained access via an Internet or other network connection from an Internet-capable appliance. Such an interface provides users with a method for storing many personal WEB pages and further provides search function and certain task-performing functions. The methods and apparatus of the present invention are taught in enabling detail below.

FIG. **1** is an overview of an Internet portal system **11** and Internet network **13** according to an embodiment of the present invention. Portal system **11**, in this embodiment, operates as an ISP in addition to a unique network portal, but may, in other embodiments be implemented as a stand-alone Internet server. In yet other embodiments the service and apparatus described herein may also be provided by such as a search and listing service (AltaVista™, Yahoo™) or by any other enterprise, hosting a WEB-connected server.

Internet **13** is representative of a preferred use of the present invention, but should not be considered limiting, as the invention could apply in other networks and combinations of networks.

ISP **15** in this embodiment comprises a server **31**, a modem bank **33**, represented here by a single modem, and a mass storage repository **29** for storing digital data. The modem bank is a convenience, as connection to the server could be by another type of network link. ISP **15**, as is typical in the art, provides Internet access services for individual subscribers. In addition to well-known Internet access services, ISP **15** also provides a unique subscription service as an Internet portal for the purpose of storing many WEB pages or destinations along with any passwords and or personal codes associated with those pages, in a manner described in more detail below. This unique portal service is provided by execution of Portal Software **35**, which is termed by the inventors the Password-All suite. The software of the invention is referred to herein both as the Portal Software, and as the Password-all software suite. Also, in much of the description below, the apparatus of the invention is referred to by the Password-All terminology, such as the Password-All Server or Password-All Portal.

ISP **15** is connected to Internet **13** as shown. Other equipment known in the art to be present and connected to a network such as Internet **13**, for example, IP data routers, data switches, gateway routers, and the like, are not illustrated here but may be assumed to be present. Access to ISP **15** is through a connection-oriented telephone system as is known in the art, or through any other Internet/WEB access connection, such as through a cable modem, special network connection (e.g. T1), ISDN, and so forth. Such connection is illustrated via access line **19** from Internet appliance **17** through modem bank **33**.

In a preferred embodiment a user has access to Internet Password-All Portal services by a user name and password as is well known in the art, which provides an individualized WEB page to the subscriber. In another embodiment wherein a user has other individuals that use his or her Internet account, then an additional password or code unique to the user may be required before access to portal **31** is granted. Such personalized Portal WEB pages may be stored in repository **29**, which may be any convenient form of mass storage.

Three Internet servers **23**, **25**, and **27**, are shown in Internet **13**, and represent Internet servers hosted by various enterprises and subscribed to by a user operating appliance **17**. For example, server **23** may be a bank server wherein interactive on-line banking and account managing may be performed. Server **25** may be an investment server wherein investment accounts may be created and managed. Server **27** may be an airline or travel server wherein flights may be booked, tickets may be purchased, and so on. In this example, all three servers are secure servers requiring user ID and password for access, but the invention is not necessarily limited to just secure services.

In a preferred embodiment of the present invention, a subscribing user operating an Internet-capable appliance, such as appliance **17**, connects to Password-All Portal system **11** hosted by ISP **15**, and thereby gains access to a

personalized, interactive WEB page, which in turn provides access to any one of a number of servers on Internet **13** such as servers **23**, **25**, and **27**, without being required to enter additional passwords or codes. In a preferred embodiment the software that enables this service is termed Password-All by the inventors. Password-All may be considered to be a software suite executing on the unique server, and in some instances also on the user's station (client). Additional interactivity provided by portal software **35** allows a connected user to search his listed pages for information associated with keywords, text strings, or the like, and allows a user to program user-defined tasks involving access and interaction with one or more Internet-connected servers such as servers **23**, **25**, and **27** according to a pre-defined time schedule. These functions are taught in enabling detail below.

FIG. **2** is an illustration of a personalized portal page as may be seen on a display monitor according to an embodiment of the present invention, provided by Password-All Portal software **35** executing on server **31**, in response to secure access by a subscriber. Page **32** presents an interactive listing **34** of user-subscribed or member WEB pages, identified in this example by URL, but which may also be identified by any convenient pseudonym, preferably descriptive, along with user name and typically encrypted password information for each page. Listed in a first column under destination, are exemplary destinations LBC.com, My Bank.com, My Stocks.com, My shopping.com, Mortgage.com, and Airline.com. These are but a few of many exemplary destinations that may be present and listed as such on page **33**. In order to view additional listings listed but not immediately viewable from within application **33**, a scroll bar **35** is provided and adapted to allow a user to scroll up or down the list to enable viewing as is known in the art.

Items listed in list **34** in this example may be considered destinations on such as servers **23**, **25**, and **27** of FIG. **1**. Typically the URL associated with an item on this list will not take a user to a server, per se, but to a page stored on a server. User names and password data associated with each item in list **34** are illustrated in respective columns labeled user name, and password, to the right of the column labeled destination. Each listing, or at least a portion of each listing, is a hyperlink invoking, when selected, the URL to that destination. In some instances a particular service may have more than one associated URL. For example, My Bank.com may have more than one URL associated for such as different accounts or businesses associated also with a single subscriber. In this case there may be a sub-listing for different destinations associated with a single higher-level listing. This expedient is not shown, but given this teaching the mechanism will be apparent to those with skill in the art.

In some embodiments one page **33** may be shared by more than one user, such as a husband and wife sharing a common account and subscription. An instance of this is illustrated herein with respect to the server labeled Mortgage.com wherein both a John and a Jane Doe are listed together under the column labeled user name. In another embodiment, a network of individuals, perhaps business owners, authorized co-workers, investment parties, or the like may share one application. In this way, system **11** may be adapted for private individuals as well as business uses.

After gaining access to application **33** which is served via Internet portal server **31** of FIG. **1**, a user may scroll, highlight, and select any URL in his or her list **34** for the purpose of navigation to that particular destination for further interaction. Application **33** already has each password and user name listed for each URL. It is not necessary, however, that the password and user name be displayed for a user or users. These may well be stored transparently in a user's profile, and invoked as needed as a user makes selections. Therefore, a user is spared the need of entering passwords and user names for any destinations enabled by list **34**. Of course, each list **34** is built, configured and maintained by a subscribing user or users, and an editing facility is also provided wherein a user may edit and update listings, including changing URL's adding and deleting listings, and the like.

In another aspect of the invention new listings for a user's profile, such as a new passthrough to a bank or other enterprise page, may be added semi-automatically as follows: Typically, when a user opens a new account with an enterprise through interaction with a WEB page hosted by the enterprise, the user is required to provide certain information, which will typically include such as the user's ID, address, e-mail account, and so forth, and typically a new user name and password to access the account. In this process the user will be interacting with the enterprise's page from his/her browser. A Password-All plug-in is provided wherein, after entering the required information for the new enterprise, the user may activate a pre-determined signal (right click, key stroke., etc.), and the Password-All suite will then enter a new passthrough in the user's Password. All profile at the Password-All Portal server.

In a related method for new entries, the enterprise hosting the Password-All Portal may, by agreement with other enterprises, provide log-in and sign-up services at the Password-All Portal, with most action transparent to the user. For example, there may be, at the Password-All Portal, a selectable browser list of cooperating enterprises, such as banks, security services, and the like, and a user having a Password-All Portal subscription and profile may select among such cooperating enterprises and open new accounts, which will simultaneously and automatically be added to the Password-All Portal page for the user and to the server hosted by the cooperating enterprise. There may be some interactivity required for different accounts, but in the main, much information from the user's profile may be used directly without being re-entered.

The inventors have anticipated that many potential users may well be suspicious of providing passwords and user names to an enterprise hosting a Password-All Portal Server executing a service like Password-All according to embodiments of the present invention. To accommodate this problem, in preferred embodiments, it is not necessary that the user provide the cleartext password to Password. All. Instead, an encrypted version of each password is provided. When a user links to his passthrough page in Password-All at the Password-All Portal server, when he/she invokes a hyperlink, the encrypted password is returned to the user's system, which then, by virtue of the kept encryption key or master password, invokes the true and necessary password for connection to the selected destination. It is thus not necessary that cleartext passwords be stored at the Password-All Portal server, where they may be vulnerable to attack from outside sources, or to perceived misuse in other ways as well.

In a related safety measure, in a preferred embodiment of the invention, a user's complete profile is never stored on a single server, but is distributed over two or more, preferably more, servers, so any problem with any one server will minimize the overall effect for any particular user.

Password-All, as described above, allows a user to access a complete list of the user's usual cyberspace destinations,

complete with necessary log-on data, stored in an encrypted fashion, so a user may simply select a destination (a hyperlink) in the Password-All list, and the user's browser then invokes the URL for the selected destination. In an added feature, Password-All may display banner ads and other types of advertisement during the navigation time between a hyperlink being invoked and the time the destination WEB page is displayed.

In yet another embodiment of the invention, a user/subscriber need not access the Password-All page to enjoy the advantages of the unique features provided. In this variation, a Plug-In is provided for the subscriber's WEB browser. If the subscriber navigates by use of the local browser to a WEB page requiring a secure log-in, such as his/her on-line banking destination, when the subscriber is presented with an input window for ID and Password, the plug in may be activated by a predetermined user input, such as a hot key or right click of the mouse device. The plug-in then accesses, transparently, the Password-All page (which may be cached at the client), and automatically accesses and provides the needed data for log-on.

In yet another aspect of the invention a search option **37** allows a user to search list **34** for specific URL's based on typed input such as keywords or the like. In some cases, the number of URL's stored in list **34** can be extensive making a search function such as function **37** an attractive option. A criteria dialog box **51** illustrated as logically separated from and below list **34** is provided and adapted to accept input for search option **37** as is known in the art. In one embodiment, search option **37** may bring up a second window wherein a dialog box such as box **51** could be located.

In another aspect of the invention the search function may also be configured in a window invoked from window **33**, and caused to search all or selected ones of listed destinations, and to return results in a manner that may be, at least to some extent, configured by a user. For example, a dialog box may be presented wherein a user may enter a search criteria, and select among all of the listed destinations. The search will then be access each of the selected destinations in turn, and the result may be presented to the user as each instance of the criteria is found, or results may be listed in a manner to be accessed after the search.

Preferably the search function is a part of the Password-All Portal software, available for all users, and may be accessed by hyperlinks in user's personal pages. In some embodiments users may create highly individualized search functions that may be stored in a manner to be usable only by the user who creates such a function.

In many aspects of the present invention, knowledge of specific WEB pages, and certain types of WEB pages, is highly desirable. In many embodiments characteristics of destination WEB pages are researched by persons (facilitators) maintaining and enhancing Password-All Portal software **35**, and many characteristics may be provided in configuration modules for users to accomplish specific tasks. In most cases these characteristics are invoked and incorporated transparent to the user.

In yet another aspect of the present invention, the Password-All suite is structured to provide periodic reports to a user, in a manner to be structured and timed by the user, through the user's profile. For example, reports of changes in account balances in bank accounts, stock purchases, stock values, total airline travel purchases, frequent-flier miles, and the like may be summarized and provided to the users in many different ways. Because the Password-All Portal server with the Password-All software site handles a broad variety of transactional traffic for a user, there is an opportunity to summarize and collect and process statistics in many useful ways. In preferred embodiments of the invention such reports may be furnished and implemented in a number of different ways, including being displayed on the user's secure personal WEB page on the Password-All Portal.

In addition to the ability of performing tasks as described above, task results including reports, and hard documents such as airline tickets may be sent over the Internet or other data packet-networks to user-defined destinations such as fax machines, connected computer nodes, e-mail servers, and other Internet-connected appliances. All tasks may be set-up and caused to run according to user-defined schedules while the user is doing something else or is otherwise riot engaged with the scheduled task.

In another embodiment of the present invention, recognizing the increasing use of the Internet for fiscal transactions, such as purchasing goods and services, a facility is provided in a user's profile to automatically track transactions made at various destinations, and to authorize payment either on a transaction-by-transaction basis, or after a session, using access to the user's bank accounts, all of which may be pre-programmed and authorized by the user.

Other functions or options illustrated as part of application **35** include a last URL option **41**, an update function **43**, and an add function **45**. Function **41** allows a user to immediately navigate to a last visited URL. Update function **43** provides a means of updating URL's for content and new address. An add function enables a user to add additional URL's to list **34**. Similarly, function **45** may also provide a means to delete entries. Other ways to add accounts are described above. It should be noted that the services provided by the unique Password-All Portal in embodiments of the present invention, and by the Password-All software suite are not limited to destinations requiring passwords and user names. The Password-All Portal and software in many embodiments may also be used to manage all of a user's bookmarks, including editing of bookmarks and the like. In this aspect, bookmarks will typically be presented in indexed, grouped, and hierarchical ways.

There are editing features provided with Password-All for adding, acquiring, deleting, and otherwise managing bookmarks. As a convenience, in many embodiments of the invention, bookmarks may be downloaded from a user's Password-All site, and loaded onto the same user's local browser. In this manner, additions and improvements in the bookmark set for a user may be used without the necessity of going to Password-All. Further, bookmarks may be uploaded from a user's local PC to his/her home page on the Password-All site by use of one or more Password-All plug-ins.

It will be apparent to the skilled artisan, given the teaching herein, that the functionality provided in various embodiments of the invention is especially applicable to Internet-capable appliances that may be limited in input capability. For example, a set-top box in a WEB TV application may well be without a keyboard for entering IDs and Passwords and the like. In practice of the present invention keyboard entry is minimized or eliminated. The same comments apply to many other sorts of Internet appliances.

In preferred embodiments of the invention, once a subscriber-user is in Password-All, only an ability to point-and-click is needed for all navigation. To get into the Password-All site, using a limited apparatus, such as an appliance without a keyboard or keypad, a Smartcard or embedded password may be used, or some other type of authentication.

It will be apparent to one with skill in the art that an interactive application such as application **33** may be provided in a form other than a WEB page without departing from the spirit and scope of the present invention. For example, an application such as application **33** may be provided as a downloadable module or program that may be set-up and configured off-line and made operational when on-line.

FIG. **3** is a flow diagram illustrating user interaction with the Internet Password-All Portal of FIG. **1**. The following process steps illustrated, according to an embodiment of the present invention, are intended to illustrate exemplary user-steps and automated software processes that may be initiated and invoked during interaction with an Internet portal of the present invention such as portal **31** of FIG. **1**. In step **53** a user connects to the Internet or another previously described switched-packet network via a compatible appliance such as Internet appliance **17** of FIG. **1**.

At step **55**, a user enters a user-name and password, which, in one embodiment, may simply be his ISP user name and password. In another embodiment, a second password or code would be required to access an Internet portal such as portal server **31** of FIG. **1** after logging onto the Internet through the ISP. In some cases, having a special arrangement with the ISP, there may be one password For both Internet access through the ISP and for Password-All. At step **57** a personal WEB page such as page **32** of FIG. **2** is displayed via Internet portal server **31**. At minimum, the personalized WEB page will contain all user configured URL's, and may also be enhanced by a search function, among other possibilities.

In step **58** a user will, minimally select a URL from his or her bookmarked destinations, and as is known by hyperlink technology, the transparent URL will be invoked, and the user will navigate to that destination for the purpose of normal user interaction. In this action, the Password-All Portal software transparently logs the user on to the destination page, if such log-on is needed.

At step **60** the user invokes a search engine by clicking on an option such as described option **37** of FIG. **2**. At step **62**, the user inputs search parameters into a provided text field such as text field **51** of FIG. **2**. After inputting such parameters, the user starts the search by a button such as button **52**. The search engine extracts information in step **64**. Such information may be, in one option, of the form of URL's fitting the description provided by search parameters. A searched list of URL's may be presented in a separate generated page in step **66** after which a user may select which URL to navigate to. In an optional search function, the user may provide search criteria, and search any or all of the possible destinations for the criteria.

In another embodiment wherein WEB pages are cached in their presentable form, information extracted in step **64** may include any information contained in any of the stored pages such as text, pictures, interactive content, or the like. In this case, one displayed result page may provide generated links to search results that include the URL associated with the results. Perhaps by clicking on a text or graphic result, the associated WEB page will be displayed for the user with the result highlighted and in view with regards to the display window.

Enhanced Agent for WEB Summaries

In another aspect of the present invention, a software agent, termed a gatherer by the inventors, is adapted to gather and return summary information about URL's according to user request or enterprise discretion. This is accomplished in embodiments of the present invention by a unique scripting and language parsing method provided by the inventor wherein human knowledge workers associated with the service provide written scripts to such a gatherer according to subscriber or enterprise directives. Such a software gatherer, and capabilities thereof, is described in enabling detail below.

Referring now to FIG. **1**, there is illustrated an exemplary architecture representing a portal service-network which, in this case is hosted by ISP **15**. Portal software **35** in this embodiment executes on portal server **31** set-up at the ISP location. Mass repository **29** is used for storing subscriber information such as passwords, log-in names, and the like. Internet servers **23**, **25**, and **27** represent servers that are adapted to serve WEB pages of enterprises patronized by a subscriber to the portal service such as one operating Internet appliance **17**.

The main purpose of portal software **35** as described above with reference to FIG. **2**, is to provide an interactive application that lists all of the subscriber's WEB sites in the form of hyperlinks. When a user invokes a hyperlink from his personal list, software **35** uses the subscriber's personal information to provide an automatic and transparent log-in function for the subscriber while jumping the subscriber to the subject destination.

Referring again to FIG. **2**, an interactive list **34** containing user-entered hyperlinks and a set of interactive tools is displayed to a subscriber by portal software **35** of FIG. **1**. One of the tools available to a subscriber interacting with list **34** is agent (software) **39**. Agent **39** may be programmed to perform certain tasks such as obtaining account information, executing simple transactions, returning user-requested notification information about upcoming events, and so on. Search function **37** and update function **43** may be integrated with agent **39** as required to aid in functionality.

It is described in the above disclosure that agent **39** may, in some embodiments, search for and return certain summary information contained on user-subscribed WEB pages, such as account summaries, order tracking information and certain other information according to user-defined parameters. This feature may be programmed by a user to work on a periodic time schedule, or on demand.

In the following disclosure, enhancements are provided to agent **39**. Such enhancements, described in detail below, may be integrated into agent **39** of portal software **35** (FIGS. **1** and **2** ); and may be provided as a separate agent or gatherer to run with portal software **35**; or may, in some embodiments, be provided as a standalone service that is separate from portal software **35**.

FIG. **4** is a block diagram illustrating a summarization software agent **67** and various capabilities and layers thereof according to an embodiment of the present invention. Summarization agent **67**, hereinafter termed gatherer **67**, is a programmable and interactive software application adapted to run on a network server. Gatherer **67** may, in one embodiment, be integrated with portal software **35** of FIG. **1** and be provided in the form of a software module separate from agent **39** (FIG. **2**). In another embodiment, gatherer **67** may be a part of agent **39** as an enhancement to the function of that agent as previously described. In still another embodiment, gatherer **67** may be provided as a parent or client-side application controlled by a separate service from the portal service described above.

In this exemplary embodiment gatherer **67** is a multi-featured software application having a variety of sub-

modules and interface modules incorporated therein to provide enhanced function. Gatherer **67** has a client/service interface layer **69** adapted to enable directive input from both a client (user) and a knowledge worker or workers associated with the service. A browser interface **77** is provided in layer **69**, and adapted to provide access to application **67** from a browser running on a client's PC or other Internet or network appliance. Interface **77** facilitates bi-directional communication with a user's browser application (not shown) for the purpose of allowing the user to input summary requests into gatherer **67** and receive summary results. Interface **77** supports all existing network communication protocols such as may be known in the art, and may be adapted to support future protocols.

Layer **69** also comprises a unique input scripting module **79** that is adapted to allow a human knowledge worker to create and supply directive scripts containing the site logic needed by gatherer **67** to find and retrieve data from a WEB site. In this case, gatherer **67** executes and runs on a network server such as server **31** of FIG. **1**. However, this is not required in order to practice the present invention.

It is assumed in this example that gatherer **67** is part of the portal software suite **35** running on server **31** of FIG. **1**. Gatherer **67** may be provided as several dedicated agents, or as one multi-functional agent without departing from the spirit and scope of the present invention. For example, one gatherer **67** may be scripted and programmed to execute a single user request with additional gatherers **67** called upon to perform additional user-requests. Alternatively, one gatherer **67** may be dedicated and assigned to each individual user and adapted to handle all requests from that user.

Interface layer **69** facilitates exchange of information from both a client and a knowledge worker. A client operating a WEB browser with an appropriate plug-in is enabled to communicate and interact with gatherer **67**. For example, a user may enter a request to return a summary of pricing for all apartments renting for under $1000.00 per month located in a given area (defined by the user) from apartments.com (one of user's registered WEB sites). The just mentioned request would be categorized as either a periodic request, or a one time (on demand) request. The communicated request initiates a service action wherein a knowledge worker associated with the service uses module **79** to set-up gatherer **67** to perform it's function. Module **79** is typically executed from a network-connected PC operated by the knowledge worker.

According to an embodiment of the present invention, a unique scripting method facilitated by module **79** is provided to enable gatherer **67** to obtain the goal information requested by a user. For example, the above mentioned example of WEB-site apartments.com has a specific HTML (hyper-text-markup-language) logic that it uses to create its site and post its information. Such site logic is relatively standard fare for a majority of different sites hosted by different entities. Using this knowledge, a knowledge worker creates a site-specific script or template for gatherer **67** to follow. Such a template contains descriptions and locations of the appropriate fields used, for example, at apartments.com. Apartment description, location, deposit information, rental information, agent contact information, and other related fields are matched in terms of location and label description on the template created with module **79**. Completed templates are stored in a database contained in a storage facility such as, perhaps, repository **29** of FIG. **1**. Such templates may be reused and may be updated (edited) with new data.

In one embodiment, one script may contain site logics for a plurality of WEB pages and instructions for specific navigational instruction and password or log-in information may be contained therein and executed serially, such as one site at a time. It is important to note that the knowledge worker or workers may perform much of their scripting via automatic controls such as by object linking and embedding (OLE) and a minor portion of scripting may be performed manually in an appropriate computer language, many of which are known in the art).

Gatherer **67** also has a process layer **71** adapted for internal information gathering and parameter configuration. An optional portal server interface **81** is provided and adapted to allow gather **67** to provide updated information to a user's list of hyperlinks and also to obtain data from portal server **31** if required. For example, required hyperlinks may be mirrored from a user's home page to a scripting template for navigational purposes. In an embodiment wherein gatherer **67** is part of a standalone service, a convention for providing user log-in information may be supplied at the client's end when a request is made. For example, an encrypted password may be supplied by a client plug-in and gatherer **67** may temporarily borrow the user's encryption key when auto log-in is performed.

An appliance configuration module **83** is provided and adapted to allow a user to define and configure an Internet appliance to communicate with the service and receive summary information. Such appliances may include but are not limited to palm top PC's, lap top PC's, cellular telephones, WEB TV's, and so on. Typically, a user will be presented a configuration WEB page from a network server that displays in his browser window on his desktop PC. The page contains an interface for communicating device parameters and communication protocol types to module **83**. In this way, a user may configure a preferred device for receipt of summary information. Device parameters and communication protocols inherent to such a device are incorporated into the scripting of the site template and are used as instructions for WEB summary delivery.

A navigation layer **73** is provided and adapted to perform the function of external site navigation and data gathering for gatherer **67**. To this end, a communication interface/browser control module **85** is provided and adapted to function as a WEB browser to access WEB sites containing WEB data. Control **85** receives it's instruction from the scripted template created by the knowledge worker.

A parsing engine **87** is provided and adapted to parse individual WEB sites according to a template created via scripting module **79**. Parsing engine **87** may be a Pearl engine, an IE HTML engine, or any other or combination of known parsing engines. The template (not shown) tells control **85** and parsing engine **87** where to go and what fields at the destination site to look for to access desired data. Once the data fields are located, parsing engine **87** gathers current data in the appropriate field, and returns that data to the service for further processing such as data conversion, compression and storage, and the like.

Because WEB sites use tools that use consistent logic in setting up their sites, this logic may be used by the summarization service to instruct control **83** and parsing engine **87**. The inventor provides herein an exemplary script logic for navigating to and garnishing data from amazon™.com. The hyperlinks and/or actual URL's required for navigation are not shown, but may be assumed to be included in the template script. In this example, a company name Yodlee (known to the inventors) is used in the script for naming object holders and object containers, which are in this case Active X™ conventions. In another embodiment, Java™

script or another object linking control may be used. The scripted template logic example is as follows:

```
# Site amazon.orders.x - shows status of orders from Amazon
login ( 7 );
get( "/exec/obdios/order-list/" );
my @tables = get_tables_containing_text( "Orders:" );
my $order_list = new Yodlee::ObjectHolder( 'orders' );
$order_list->source( 'amazon' );
$order_list->link_info( get_link_info() );
my @href_list;
my @container_list;
foreach my $table ( @tables ) {
        my @rows = get_table_rows();
foreach my $i (0 . . $#rows ) {
        select_row ( $i );
        my $text = get_text( $rows[ $i ] );
        next if $text =~ /Orders:|Status/;
        my @items = get_row_items();
        next unless @items >= 4;
        my( $order_num, $date, $status );
        select_cell( 1 );
        $order_num = get_cell_text();
        my $href = get_url_of_first_href( get_cell() );
        select_cell( 2 );
        $date = get_cell_text();
        select_cell( 3 );
        $status = get_cell_text();
        next unless defined $order_num and defined $date and
        defined
$status;
        $order = new Yodlee::Container( 'orders' );
        $order->order_number( $order_num );
        $order->date( $date );
        $order->status( $status );
        $order_list->push_object( $order );
        if( defined $href ) {
          push( @href_list, $href );
          push( @container_list, $order );
foreach my $i ( 0 . . $#href_list ) {
        get( $href_list[ $i ] );
        @tables = get_tables_containing_text( "Items
        Ordered:" );
foreach my $table ( @tables ) {
        my @rows = get_table_rows();
foreach my $j ( 0 . . $#rows ) {
        select_row( $j );
        my $href = get_url_of_first_href( get_row() );
        next unless defined $href;
        my @child_list = get_children( get_row(), 'a' );
        next unless defined $child_list[ 0 ];
        my $text = get_text( $child_list[ 0 ] );
        $container_list[ $i ]->description( $text );
    }
  }
}
result( $order_list );
```

The above example is a script that instructs control **85** and parser **87** to navigate to and obtain data from Amazon™.com, specifically that data that reflects the user's current order status. Scripts may also be written to obtain virtually any type of text information available from any site. For example, a user may wish to obtain the New York Times headlines, the top ten performing stocks, a comparative list of flights from San Francisco to New York, etc. In one embodiment, metadata may be associated with and used in-place of the actual scripted language for the purpose of reducing complication in the case of many scripts on one template.

A data processing layer **75** is provided and adapted to store, process, and present returned data to users according to enterprise rules and client direction. A database interface module **89** is provided and adapted to provide access for gatherer **67** to a mass repository such as repository **29** of FIG. **1**, for the purpose of storing and retrieving summary data, templates, presentation directives, and so on. Gatherer agent **67** may also access data through interface **89** such as profile information, user account and URL information, stored site logics and so on. Data scanned from the WEB is stored in a canonical format in a database such as repository **29**, or in another connected storage facility. All stored data is, of course, associated with an individual who requested it, or for whom the data is made available according to enterprise discretion.

A summarization page module **91** is provided and adapted to organize and serve a WEB summary page to a user. Module **91**, in some embodiments, may immediately push a WEB summary to a user, or module **91** may store such summarized pages for a user to access via a pull method, in which case a notification may be sent to the user alerting him of the summary page availability. Summarization module **91** includes an HTML renderer that is able to format data into HTML format for WEB page display. In this way, e-mail messages and the like may be presented as HTML text on a user's summarization page. Moreover, any summary data from any site may include an embedded hyperlink to that site. In this way, a user looking at an e-mail text in HTML may click on it and launch the appropriate e-mail program. Other sites will, by default, be linked through the summary page.

Many users will access their summary data through a WEB page as described above, however, this is not required in order to practice the present invention. In some embodiments, users will want their summary information formatted and delivered to one of a variety of Internet-capable appliances such as a palm top or, perhaps a cell phone. To this end, the renderer is capable of formatting and presenting the summary data into a number of formats specific to alternative devices. Examples of different known formats include, but are not limited to XML, plain text, VoxML, HDML, audio, video, and so on.

In a preferred embodiment of the present invention, gather **67** is flexible in such a way as it may act according to enterprise rules, client directives, or a combination of the two. For example, if a user makes a request for summary data about a user/subscribed WEB page to be periodically executed and presented in the form of a HTML document, then gather **67** would automatically access and analyze the required internal information and user provided information to formulate a directive. Using scripting module **79**, a knowledge worker provides a template (if one is not already created for that site) that contains the "where to go" and "what to get" information according to site logic, user input, and known information.

Alternatively, if a user requests a summary about data on one of his sites such as, perhaps, current interest rates and re-finance costs at his mortgage site, the service may at it's own discretion provide an additional unsolicited summary from an alternate mortgage site for comparison. This type of summarization would be designed to enhance a user's position based on his profile information. In this case, updated data about latest interest rates, stock performances, car prices, airline ticket discounts, and so on would be stored by the service for comparative purposes. If a user request for a summary can be equaled or bettered in terms of any advantage to the user, such summary data may be included.

In many cases, created templates may be re-used unless a WEB site changes it's site logic parameters, in which case, the new logic must be accessed and any existing templates must be updated, or a new template may be created for the site. The templates contain site-specific script obtained from

the site and stored by the knowledge workers. In one embodiment, companies hosting WEB pages automatically provide their site logics and any logic updates to the service by virtue of an agreement between the service and the WEB hosts.

In an alternative embodiment gatherer **67** may be implemented as a client application installed on a user's PC. In this embodiment, a user would not be required to supply log-in or password codes. Summarization scripts may be sent to the client software and templates may be automatically created with the appropriate scripts using log-in and password information encrypted and stored locally on the user's machine.

In addition to providing WEB summary information, gatherer **67** may also be used to provide such as automatic registration to new sites, and for updating old registration information to existing sites. For example, if a user whishes to subscribe, or register at a new site, only the identification of the site is required from the user as long as his pertinate information has not changed. If a new password or the like is required, gatherer **67** through control module **73** may present log-in or password codes from a list of alternative codes provided by a user. In another embodiment, a database (not shown) containing a wealth of password options may be accessed by gatherer **67** for the purpose of trying different passwords until one is accepted by the site. Once a password or log-in code is accepted, it may be sent to a user and stored in his password list and at the network level.

It will be apparent to one with skill in the art that a software application such as gatherer **67** may be implemented in many separate locations connected in a data network. For example, a plurality of gatherer applications may be distributed over many separate servers linked to one or more mass repositories. Client applications include but are not limited to a WEB-browser plug-in for communicating to the service. Plug-in extensions may also be afforded to proxy servers so that auto-log-in and data access may still be performed transparent to a user.

In another embodiment, plug-ins enabling communication with gatherer **67** may be provided and configured to run on other network devices for the purpose of enabling such a device to initiate a request and get a response without the need for a desktop computer.

In most embodiments a user operating a desktop PC will order a one time or periodic summary related to some or all of his subscribed WEB sites. A logical flow of an exemplary request/response interaction is provided below.

FIG. **5** is a logical flow chart illustrating an exemplary summarization process performed by the software agent of FIG. **4** operating in a user-defined mode. In step **93**, a user has initiated a new request for a summary (summary order). It is assumed for the purpose of discussion, that the request of step **93** involves a site wherein no template has been created. In step **95**, the request is received and analyzed. A knowledge worker will likely perform this step. The new request may be posted to the user's portal home page, sent directly to gatherer **67**, or even communicated through e-mail or other media to the service.

In step **97** a knowledge worker accesses particular site logic associated with the request URL'S. For example, if the request involves a plurality of URL's, then all site logics for those URL's are accessed. Logic may be available in a repository such as repository **29** of FIG. **1** if they were obtained at the time of user registration to a particular URL, or sent in by WEB-site hosts shortly after registration. If it is a completely new URL, then the logic must be obtained from the site. In most cases however, the logic will be known by virtue of a plurality of users accessing common URL's. Therefore cross-linking in a database of logic/user associations may be performed to access a logic for a site that is new to one particular user, but not new to another.

In step **99**, the knowledge worker creates a template by virtue of scripting module **79** (FIG. **4**) containing all site logic, URL's, log-in and password information, and the user request information. As described previously, templates may be re-used for a same request. In most cases, scripting may be mostly automated with minimum manual input performed by the knowledge worker. In many cases, an existing template will match a new request exactly, and may be re-used. In that case steps **97**, **99**, and **101** would not be required.

In step **101** the template is stored and associated with the requesting user. The stored template may now be retrieved at a scheduled time for performing the summary gathering. At step **103**, a browser control such as module **85** of FIG. **4** is activated to access the stored template and navigate to specified URL's for the purpose of gathering summary data. If a timing function is attributed to the template stored in step **101**, then the template may self execute and call up the browser function. In another embodiment, the knowledge worker may notify the browser control to get the template for it's next task. In some embodiments, a plurality of controls may be used with one template as previously described.

In step **105**, automatic log-in is performed, if required, to gain access to each specified URL. In step **107**, a specified WEB-page is navigated to and parsed for requested data according to the logic on the template. If there are a plurality of WEB-pages to parse, then this step is repeated for the number of pages. A variety of parsing engines may be used for this process such as an IE™ parser, or a Pearl™ parser. Only the requested data is kept in step **107**.

A request may be an on-demand request requiring immediate return, or a scheduled request wherein data may be posted. At step **109**, such logic is confirmed. If the data is to be presented according to a periodic schedule, then summary data parsed in step **107** is stored for latter use in step **111**. In step **113**, the summary data is rendered as HTML if not already formatted, and displayed in the form of a summary WEB-page in step **115**. The summary page may be posted for access by a user at a time convenient to the user (pull), or may be pushed as a WEB-page to the user and be made to automatically display on the user's PC. Notification of summary page availability may also be sent to a user to alert him of completion of order.

If the summary data is from a one-time on-demand request and required immediately by a user, then a network appliance and data delivery method (configured by the user) is confirmed, and the data is rendered in the appropriate format for delivery and display in step **117**. In step **119**, the summary data is delivered according to protocol to a user's designated appliance. In step **121** a user receives requested information in the appropriate format.

It will be apparent to one with skill in the art that there may be more or fewer logical steps as well as added sub-steps than are illustrated in this example. For example, step **105** may in other embodiments include sub-steps such as getting an encryption key from a user. In still another embodiment, part of a request may be rendered as HTML as in step **113** while certain other portions of the same request data might be rendered in another format and delivered via alternative methods. There are many possibilities.

The method and apparatus of the present invention may be used to present summaries to users without user input. Process logic such as this is detailed below.

FIG. **6** is a logical flow chart illustrating an exemplary summarization process performed by the software agent of FIG. **4** in a User-independent smart mode with minimum or no user input. In step **117** an enterprise-initiated summary process begins. In this case, the enterprise may be assisting a user in finding a better deal or, perhaps presenting the individual with summaries from and links to alternative pages not yet subscribed to by a user.

In step **119**, a database containing user information and parameters is accessed and reviewed. Certain information specific to a user may be required to initiate an enterprise-sponsored summary report. At step **121**, the knowledge worker accesses the site logic specific to the specified target site or sites for summarization. In step **123**, the knowledge worker modifies an existing user template, or creates a new one if necessary. At step **125** the template is stored in a repository such as repository **29** and associated with the user.

As described in FIG. **5**, the template either self-executes according to a timed function and invokes a browser control such as control **85** (FIG. **4**), or is accessed by control **85** as a result of task notification. In step **127**, the browser control begins navigation. Auto log-ins are performed, if required, in step **129** to gain access to selected sites. If the WEB pages are new to a user, and the user has no registration with the WEB site, then through agreement, or other convention, the service may be provided access to such sites. Such an agreement may be made, for example, if the host of the WEB site realizes a possibility of gaining a new customer if the customer likes the summary information presented. In many other situations, no password or log-in information is required to obtain general information that is not personal to a client.

In step **131**, all sites are parsed for summary data and stored in canonical fashion in step **133**. At step **135**, the data is compiled and rendered as HTML for presentation on a summary page. In step **137**, a WEB summary containing all of the data is made available to a user and the user is notified of it's existence.

Providing certain information not requested by a user may aid in enhancing a user's organization of is current business on the WEB. Moreover, unsolicited WEB summaries may provide better opportunities than the current options in the user's profile. Of course, assisting a user in this manner will require that the enterprise (service) have access to the user's profile and existing account and service information with various WEB sites on the user's list. A user may forbid use of a user's personal information, in which case, no enterprise-initiated summaries would be performed unless they are conducted strictly in an offer mode instead of a comparative mode.

The method and apparatus also may be practiced in a language and platform independent manner, and be implemented over a variety of scalable server architectures.

Deeper-Level Searching by Proxy

As described in the background section, a conventional search function cannot search beyond a first level of WEB-site depth. A URL must be pre-known either to a user or to a service providing data-search capability before it may be returned as a result of a search. Vast numbers of URLs are not indexed into any search engine databases and therefore cannot be found by traditional key-word searching. An overview of prior art implementation of a traditional data-search process as practiced on the Internet is provided below.

FIG. **7** is an architectural overview of a system employing a conventional data-search process for on a DPN network according to prior art. A communication network **139** is exemplified in this prior-art example as a common architecture for facilitating network data searches. Network **9** comprises the well-known Internet network **141**, a well-known public-switched-telephony network (PSTN) **143**, an Internet Service Provider (ISP) **145**, and an exemplary user premise **147**.

It is widely known and accepted in the art that Internet **141**, PSTN **143**, ISP **145**, and user premise **147** represent a communication architecture (network **139**) commonly used by the public for searching out and obtaining network-sourced data.

Internet **141** has an Internet backbone **157** illustrated therein and intended to represent the many lines and connection points making up the Internet network as a whole. Two search provider (SP) servers, server **149** and server **151** are illustrated as connected to backbone **157**, and are adapted to provide Internet data-search services to the public at large as is generally known in the art. A search provider is defined, for the purpose of this example, as an enterprise engaging in providing WEB-sourced data made accessible to users through server capabilities. Altavista™ and Yahoo™ are well-known examples of search providers Such enterprises may also provide other services such as portal services and so forth.

SP server **149** has a data store **153** connected thereto by a data link. Data store **153** is adapted to contain cached URLs, which are compiled and indexed according to enterprise rules and which are accessible through a search-engine application illustrated as SW **163** running on server **149**. Data store **153** may be any kind of suitable data repository capable of storing large amounts of data. Data store **153** is typically an on-line data repository which is accessed by server **149** when matching data-search queries to data contained in data store **153**.

SP server **151**, a connected data store **155**, and an instance of SW **165** may be described as replicated components of server **149**, data store **153**, and SW **163** accept that a differing, enterprises may host such services. For example, Altavista™ may host server **149**, data store **153**, and SW **163** while Excite™ may host server **151**, data store **155** and SW **165**. Slight differences may exist between the separate enterprises hosting the aforementioned equipment. Therefore, physical differences may exist in the services offered as well as in SW and hardware implementations. The inventor chooses to focus only on the standard data-search functionality common to both equipment and SW groups. Therefore, each group is represented with identical capabilities in this example.

Two WEB servers (WS), **159** and **161** are illustrated as connected to backbone **157** in Internet **141**. WSs **1.59** and **161** are adapted as normal file servers as known in the art. Servers **159** and **161** host electronic information pages addressed by URLs and are adapted to serve them on authorized request from any other network-connected node. Electronic WEB-pages are typically formatted in well-known Hyper-Text-Mark-up Language (HTML). The URL is actually the unique server address of an information page as is well known.

PSTN **143** represents the most common telephony network used to access Internet **141**. PSTN **143** may be assumed to contain all of the required equipment for enabling telephony communication and connection including such as telephony switches, routers, service control points (SCP), network bridging stations, and so on.

ISP **145** is provided within PSTN **143** and is adapted to perform Internet-access services as known in the art. ISP **145** comprises a modem bank **171**, represented herein by a single modem icon, and an Internet connection server **169** adapted to connect subscribers to Internet **141**. Connection server **169** is illustrated as having connection to Internet backbone **157** by an Internet access line **167**. Access line **167** may be any suitable connection means known in the art for maintaining Internet connectivity for a plurality of users accessing Internet **141** through server **169**.

User premise **147** comprises a personal computer (PC) **175**, which is adapted by SW and hardware implementation for communication on Internet **141**. PC **175** is illustrated as connected to modem bank **171** by an Internet access line **173**, which may be any connection means known in the art for providing Internet access to user premise **147**. Examples include normal plain old telephone service (POTS) line, Integrated Services Digital Network (ISDN) line, Cable/Modem line and so on. In this example, PC **175** uses a dial-up method and ISP **145** to access Internet **141** as is most common in the art.

A browser application **177** is provided and illustrated as executing on PC **175** indicating that PC **175** is engaged in a browsing session on Internet **141**. A search engine, represented within browser **177** by the letters SE is incorporated by a user operating PC **175** for the purpose of data search as is known in the art. A user operating PC **175** and connected to Internet **141** through ISP **145**, as illustrated by the described connections, may invoke an SE through application **177** and thus connect to one of SP servers **149** or **151** in Internet **141**. The exact server connection will depend on the proprietary search option listed in application **177** and selected by a user. Using the examples presented above, if the search option chosen is Altavista™, then PC **175** will be connected to SP server **149** hosted by Altavista™. If the chosen option is Excite™, then PC **175** would be connected to SP server **151** hosted by Excite™. Such methods are known in the art and many different search providers hosting separate data services may be represented for selection in application **177**.

Assuming that a user operating PC **147** is connected to Internet **141** through one of several methods provided as examples above, a data search may be initiated from application **177** by invocation of search option SE provided as a link in application **177**. Assuming that upon invocation of SE in application **177**, a connection to SP server **149** is made, then an interactive HTML page representing a data-search interface is served to the connected user. SW **163** running on server **149** then processes any initiated data search according to a query entered into a search dialog box provided with the HTML interface as is known in the art.

In process of a query from a user operating PC **175**, SP server **149** running SW **163** checks data store **153** for any URL pages contained therein that have data content associated therewith that matches (to some extent) criteria according to the entered query. As described in the background section, a query may be a key word, a series of key words, a phrase or the like. Server **149** running SW **163** returns any matching URL's from data store **153**, where they appear in listed fashion in application **177**. URL results are often termed "hits" in the art. There may be only a few or a great number of "hits" returned depending on the nature (broad or narrow) of the original query entered, and the richness of the Internet content. Each hit represents a hyper-link to an electronic WEB page that may be hosted, in this example, by server **159** or server **161**, or any other network-connected server. Therefore, invoking a returned URL initiates navigation by browser **177** to either server **159** or **161** wherein the updated version of the HTML page is served. At this point the aforementioned user is negotiating with server **159** or **161**.

There are other possible aspects of connection and communication represented in this prior-art example as well. For example, an enterprise hosting SP server **149** may through agreement forward a query to the enterprise hosting SP server **151** such that data store **155** may be included in a data search. With this type of cooperation, many resources may be accessed in a shared sense. Therefore, if an original query does not return a URL from one data store, an option may exist for searching data stores hosted by other enterprises without a user having to close one connection and open another. This process is fairly recent and is termed meta-searching in the art.

A limitation of the prior art exists in that software instances **163** and **165** are adapted only to provide URL's and data that is indexed in either data store **153** or data store **155**. An enterprise hosting server **159** or server **161** may also have connected data-stores containing information related to electronic pages that are hosted therein. Such data stores hold data on a deeper level of WEB-site depth and may be accessed through manual navigation from a main URL or through a private search function (limited to searching data hosted by the enterprise) provided as an embedded module in one or more of the hosted main pages. Software instances **163** and **165** cannot provide access to a private search function unless it is functionally available in either server **149** or server **151**. A user must invoke the private search function after he or she is served the hosting page in order to search a private data-store. Moreover, a user must often restructure a query for application to the new search engine soilware as the query rules may be different than those associated with SW **163** or SW **165**.

It can be seen, in a general sense, by one with skill in the art that the prior-art data search methods illustrated in this example are limited both by the fact that only data indexed by URL may be found, and by the fact that additional deeper-level data searches must be performed manually through user-initiated browser navigation.

The inventor provides a unique method and apparatus that enables a deeper-level data search to be accomplished through an original SE application wherein no query re-entering by a user or additional browser navigation by a user is required. Such a method and apparatus is described in enabling detail in examples below.

FIG. **8** is an architectural overview of a search method for data on Internet **141** according to an embodiment of the present invention. Much of the architecture and connection means illustrated in this preferred embodiment mirror those of the prior-art example of FIG. **7**. Therefore, elements common to both examples retain the same element numbers and are not re-introduced. Components unique to the present invention whether by modification or by provision are newly introduced and given new element numbers.

In this example of the present invention, the Internet connection means is the same as described in FIG. **8** above. A user operating PC **175** is connected to modem bank **171**, hosted by ISP **145**, by virtue of Internet access line **173**. Connection server **169**, also hosted by ISP **145**, facilitates connection to Internet backbone **157** within Internet **141** through Internet access line **167**. However, instead of using a general search engine as was illustrated in FIG. **8**, a user operating PC **175** is a subscriber to the personalized portal service described in disclosure included herein and refer-

enced as Ser. No. 09/208,740 in the cross-reference section above. As such, a connection is opened to a portal server **179** upon Internet log-in from user premise **147** and a portal page illustrated as PP is served by server **179** and appears within a browser application **178**.

Browser application **178** is enhanced for communication with portal server **179** by virtue of provided SW plug-ins (not shown), which are adapted for enabling auto-log-in to personal WEB pages, initiating special tasks to be performed by server **179**, among other options which are fully described in the related documents Ser. No. 09/523,598 and Ser. No. 09/208,740. A user operating PC **175** while connected on-line to portal server **179** may interact with the provided PP in browser **178** to search for updated data from one or all of his or her service-registered WEB pages. In this system, portal server **179** is enhanced with a navigation control for browsing on behalf of a user operating PC **175**. In general, such navigation and return of data is limited to sites that are known to the service and/or to the user. For example, navigation to sites for data acquisition on behalf of a user is accomplished with site-logic scripting, parsing and data-return techniques known to the inventor and described above. The portal service uses a system of connected nodes to process the many requests from users.

A data store **185** is provided and illustrated as connected to portal server **179** by data link. Data store **185** is adapted to contain and manage data including but not limited to profile and subscription data about users, data about user-registered sites, password and user-names associated with those sites, and navigation scripts for accessing such sites on behalf of users. Data store **185** may be a series of separate data repositories all connected to server **179**, or a single repository as represented herein, or a part of portal server **179**. Data store **185** may be of any suitable implementation such as an optical storage facility or the like. In this example, server **179** and connected data store **185** are held within Internet **141** with server **179** directly connected to backbone **157**. However, in another embodiment, server **179** and data store **185** may be hosted by and held within ISP **145** as represented in FIG. **1**.

Three WEB servers (WS) **181***a–c* are illustrated as connected to backbone **157** in Internet **141**. WEB servers **181***a–c* are adapted as Internet file servers as described in FIG. **7** (WS **159**, WS **161** ). However, in this embodiment each WS **181***a–c* has at least one main HTML page hosted therein that contains a private search engine (SE) embedded therein as illustrated by associated flags labeled SEa, SEb, and SEc respectively.

An on-line database **187** is provided and illustrated as connected to backbone **157** within Internet **141**. Database **187** represents an on-line storage facility containing additional HTML pages hosted by WEB servers **181***a–c*. Database **187** may be a single data repository shared by servers **181***a–c* as is represented herein or database **187** may represent a separate database for each of WEB servers **181***a–c*. Database **187** stores electronic WEB pages that may be accessed through a private SE hosted in any one of or all of servers **181***a–c*. For example, WS **181***a* may be hosted by Intel™. As such, electronic pages contained in database **187** represent deeper-level electronic pages containing information related to Intel™ and accessible through SEa hosted at server **181***a*, but not indexed by a regular SE database such as, perhaps, Altavista™. WS **181***b* may be hosted by Gateway™ and an embedded SEb, also hosted by Gateway™ may be used to search database **187** for URLs related to Gateway™ such as computer specifications, chip parameters, install instructions, and so on.

It is important to note here that pages having URLs maintained in database **187** cannot typically be accessed through a conventional search method because they represent a deeper level of WEB data not indexed in either data store **153** or data store **155** of FIG. **7**. The additional pages are only accessible through use of embedded SE applications found on such as a main electronic page or pages hosted in servers **181***a–c*, or through manual navigation from one of the main URLs providing links to the deeper-level information. A private SE may be a search function dedicated to providing access to additional technical service-related URLs hosted by an enterprise. The specific SE may be labeled "search our technical service site", for example, and may be configured to search by key word or phrase. The search provided is, of course, limited to enterprise-hosted databases such as database **187**.

In a conventional sense (negotiaiting with the server hosting the SE), one would enter a key word or the like into the private SE as described above and would be presented with a list of hyper-links to the additional pages hosted by the enterprise which would appear in a user's browser application. The additional URLs may also be linked by icons found in various electronic pages contained in servers **181***a–c* and hosted by the respective enterprises. The use of a private SE of the type described herein allows faster access to data and reduced manual navigation time for users.

The inventor herein teaches and provides a unique application extension that enables a seamless bridge between a conventional SE and a private SE. A SW application **183**, illustrated as executing on portal server **179**, provides such enhanced functionality. In this example, SW **183** is a personalized search function provided by the enterprise hosting server **179** and the portal service, which is available to users typically through subscription. SW **183** may be invoked by a user operating PC **175** at user premise **147** by clicking on an available link presented in a PP (Portal Page) within browser application **178**.

Once SW **183** is invoked, a user operating through interface **178** enters a natural language query designed to search for specific data. It is assumed in this example that specific data requested is not contained in any of the URLs for pages registered with the portal service. It is also assumed that the requested data is available in a deeper level of data which may be accessed through use of one or more private SEs hosted by one or more of the user's registered WEB services.

To further illustrate, consider that WS **181***a* is a Hewlett Packard™ server registered to the portal service by a user operating PC **175**. PC **175**, in this example, may be a Hewlett Packard™ machine such as a Pavilion™ model machine. A query entered into a PP search dialog box may be, for example, "Bios flash upgrade information for Pavilion". SW **183** parses the entered query and processes the query by checking data store **185** for any related data. It is found that WS **181***a* (Hewlett Packard™) is a user-registered WEB site and is a likely URL for containing data related to the query. In one embodiment, a user may make a registered URL an integral part of a query command. For example, the query may read "search my HP WEB site for 'Bios flash upgrade information for Pavilion'". The double quotations illustrated in the command query may be used to separate the command portion from the query portion although this is merely exemplary. There are many ways to express command/query combinations.

SW **183** uses a navigation sub-system (not shown), which is known to the inventor, to navigate to HP server **181***a* on

the user's behalf and perform auto-log-in to access a main URL contained in server **181***a* to which the user subscribes. The requested information is not contained in the main URL, but may be available through a private SE embedded in the main URL at server **181***a* (illustrated by flag SEa). SW **183** is, in one embodiment, adapted to recognize the code that identifies the embedded SE and is adapted by software routine to locate and invoke the private SE at the main URL in server **181***a*. In another embodiment, the private SE parameters such as data entry rules are pre-known and are accessible from data store **185**.

Once the private SEa is open, SW **183** transfers the original query into the dialog box provided and executes the search function by virtue of automated routine. If required, SW **183** may restructure the query to fit the rules used by the private SE. Data returned by the private SE is gathered by a navigation control and returned to server **179** where it may be forwarded to the portal page (labeled PP in FIG. **8**) in browser interface **178**. A user may then click on any additional URL listed and navigate to that electronic page hosted in this case at database **187**, and view the data.

The search, navigation, and data-return process is transparent to the requesting user as is the auto-log-in process. The next page the user sees is a list of related links to data about "Bios flash upgrade instruction". In some cases, the additional links may appear on the same PP within browser **178** by virtue of an automated linking process known in the art. By clicking on any one of the provided links, a user may navigate to the selected page and view the data contained therein.

SW **183** thus provides a proxy searching function that may be practiced by a user from a single interface and using an original query typed into a first search dialog box. A user practicing this method is not required to manually navigate until he or she is presented with a list of links related to the deeper level data held in database **187** in this example.

It will be apparent to one with skill in the art that the functionality of SW **183** is in part generic to and in accordance with similar capabilities described in the related documents listed under the cross-reference section. Additional components added to SW **183**, which provide a novel interface capability between SE applications are detailed further below.

FIG. **9** is a block diagram illustrating exemplary software components of a search-function interface according to an embodiment of the present invention. SW **183** comprises a data-search module **184** and an application-extension layer **186**. Search module **184** is similar in many respects to traditional search engines except for the presence of a browser control interface **195**, and an interface to auto-log-in function **197**.

Control interface **195** is provided and adapted as an enhancement that allows interface to a navigation system for browsing known URLs on behalf of users. Interface **197** is provided and adapted to allow auto-log-in functions to be performed on behalf of a user upon navigation to a user-registered URL for the purpose of obtaining data requested by a user.

An input module **189** is provided and adapted to accept query data input into SW **183** by interfacing users. A parsing engine **191** is provided and adapted to read and understand data queries for purpose of further processing data requests. A database interface module **193** is provided and adapted to allow interface to any connected repository to search for data that may be compared against a query for match. Browser control **195**, as previously described, is an interface to a proxy-navigation system. If data matching a query is not found in a connected database, then navigation may be required to obtain the requested data. Auto-log-in services may be performed during navigation to gain access to user-registered sites.

Search application **184**, as known to the inventor, is not the same as a traditional search engine used for generic data searches on the Internet. Application **184** is enhanced for integration into the Password-all software suite described in Ser. No. 09/208,740 and the method for obtaining and presenting WEB summaries described in Ser. No. 09/523,598. A basic example of using search application **184** is described in the embodiment of FIG. **5** above. In this embodiment, Auto-log-in is performed during navigation to gain access to user-registered sites, which require a user name and/or password for authentication. Data is found through parsing and site logic scripting. The function of search application **184** assumes that there is sufficient pre-known information available about the data source and data location in the source for successful navigation and parsing.

Application extension **186** is provided to extend the function of application **183** to provided a seamless interface to a second search application which may be specific to an enterprise hosting a WEB site comprising am plurality of pages having URLs. Application **186** enables SW **183**, in cooperation with a proxy-navigation system, to navigate to and commandeer the second search engine and cause that engine to search for and return data on behalf of a user.

A code recognition module **199** is provided and adapted to recognize an embedded search function held within a URL opened during proxy navigation. In this way, SW **183** may find any second search function embedded in any URLs subject to navigation and search. In one embodiment, such search functions are pre-located when a user registers a new URL to the service such that their parameters and location may be made part of site-logic scripting.

An application-activation module **201** is provided within extension layer **186** and adapted to invoke or activate an embedded search function. In some cases an embedded search function on will be presented in the form of an icon such that when invoked, a dialog box appears as a pop-up widow or as a new URL. In some cases, a dialog box will already be present and module **201** may not be required.

A text writer **203** is provided and adapted to rewrite an original query into a form accepted by the search dialog criteria associated with the second search function. If required, writer **203** may restructure an original query to fit the new criteria in terms of punctuation, casing, order of words, association of words, and so on. In a Preferred embodiment, such rules are pre-known and are a part of site logic. In an alternate embodiment, writer **203** simply produces the original query for insertion into the dialog box wherein no restructuring is required.

A data-transfer interface **205** is provided and adapted to allow SW **183** to insert an original query into a provided dialog box by known techniques such as object linking and embedding (OLE). An execution and release module **207** is provided and adapted to execute a second search function after a query has been entered. At this point, the data search function is turned over to the new search function, which returns results back to the proxy navigation control. Application extension **186** actively runs in conjunction with the navigation system in integrated fashion to achieve the main object of the present invention, which is to enable a seamless interface between search applications such that a deeper level of data searching may be achieved.

Data returned by the second search function invoked by SW **183** is handled in the same way as described in FIG. **5** steps **111**, **113**, **115**, and steps **117**, **119**, and **121**. Automatic linking capability allows a user receiving requested data links to navigate back to data contained therein. In some cases data located will be returned as text data with no linking required.

It will be apparent to one with, skill in the art that the software components included in SW **183** may be provided to coordinate through interface with a separate proxy navigation system as known to the inventor, or may be functionally provided within the navigation software itself without departing from the spirit and scope of the present invention. In a preferred embodiment, the components described above are Java-based executables designed to function as a routine during Internet navigation.

The method and apparatus of the present invention provides a unique way for users to gain information by proxy from deeper levels of WEB sites without requiring exhaustive manual navigation and repeated re-entering of queries to new search functions.

In one embodiment of the present invention, more than one secondary search function, perhaps associated with more than one URL may be invoked simultaneously such that data returned to the gathering agent is from several different sources or sites.

FIG. **10** is a process flow diagram illustrating basic interaction steps for practicing the present invention according to a preferred embodiment. At step **209**, a user begins an on-line session with a portal server as exemplified in FIG. **8**. During this process, a user-name and password pair is submitted to a portal server by a user for authentication purposes. After authentication of a user, a personal portal page (PP of FIG. **8**) is displayed in a user's WEB browser at step **211**. In this step, a dialog box for SW **183** will appear in some convenient location on the portal page.

At step **215**, a user enters a query for a data search. The query may be entered in a natural language as previously described in the example of FIG. **8**. At step **213**, SW **183** processes the query for a WEB search. During this process, any connected databases are consulted for matching data before navigation is initiated. If the required data is contained in a connected database, navigation and proxy searching may not be required. For example, if a user requests data about "technical specifications for white diamonds", then a first "look" into a database may return a user-registered site about diamonds and other minerals. The URL would match the user's query but the exact data may not be found on the URL page.

Assuming that no matching data is found, navigation to the related URL is initiated through browser control interface at step **217**. Proxy navigation to the URL or URLs that most closely relate to a user query is performed by a navigation sub-system. Auto-log-in is performed if required for entry into a site.

At step **219**, any private search functions associated with the site and available on the main URL page or pages are located and invoked. At step **221**, original query data entered at step **215** is transferred to a new dialog box associated with a new search function. At this point, the search is handed over to the respective WEB site or sites. At step **223**, data results from the secondary search, which may be in the form of text, additional URL links, or a combination thereof, are passed back to the navigation control. These results represent data that could not have been obtained through conventional search methods because such methods are limited to a first WEB-site depth.

If a user requires immediate data return, the results are passed back to the user's WEB browser at step **225**. If a user will access the results at a later date, then the results may be held in storage on behalf of the user at step **227**.

It will be apparent to one with skill in the art that the basic process-interaction steps represented herein may be expanded in description without departing from the spirit and scope of the present invention. For example, step **209** may include sub-steps such as supplying password and user name for authenticating. A step for invoking an original search application may be provided between steps **211** and **215** if an open dialog box does not appear with the served portal page (PP). There are many possibilities. The inventor intends that the process steps represented herein are only exemplary of one suitable process among many for practicing the present invention.

It will also be apparent to one with skill in the art that SW **183** of FIG. **9** may be a standalone application with appropriate interface capability to a navigation sub-routine without departing from the spirit and scope of the present invention. In still another embodiment, application **183** may be integrated with a navigation sub-routine such that navigation capability is part of the direct functioning of SW **183**.

The method and apparatus of the present invention may be practiced in a personalized sense as is described in previous embodiments wherein URLs are registered to users and auto-log-in services are performed on behalf of users subscribing to portal services.

In another embodiment, the method of the present invention including proxy navigation capabilities may be provided as an extension to existing and well-known search engines that are provided to the public without subscription. Such search engines are typically used to search for more generalized data, and users do not have pre-knowledge of where requested data is held. A general search engine executing from a server may, if enhanced with the SW of the present invention, provide a deeper level of data searching than is currently offered. Such an embodiment is detailed below.

FIG. **11** is a block diagram illustrating the standard data-search system of FIG. **7** enhanced with the method and apparatus of the present invention according to another embodiment of the present invention. In this example, user premise **147** is enabled by virtue of browser application **177** to browse the Internet as described in FIG. **7**. A standard search engine is illustrated within browser **177** and is an interface to search-provider (SP server) **151**. User premise **147** has connection to server **151** by virtue of an ISP-brokered network-connection illustrated herein by the double arrow labeled "Network Connection (ISP)". Such a connection is analogous to the compilation of lines **173** and **167** of FIG. **7**.

Server **151** has an enhanced search engine **229** executing thereon and adapted to allow added services according to an embodiment of the present invention. For example, engine **229** is enhanced with addition of a proxy browsing control **195**, which allows interface to a general version of the personalized navigation system described above. What is meant by "a version of" is that no site logic is employed to look for specific data known to exist.

Search engine **229** is also enhanced with a "generalized version" of the personalized application **186** of FIG. **9**. Meaning that there is no interface for auto-log-in. Application control **195** and extension **186** may be provided with a navigation sub-routine and integrated into a standard search engine (SE) producing the enhanced engine **229**.

Alternatively, engine **229** has a navigation control or interface to a separate browsing sub-system, which may run on the same server (**151**), or another connected server or set of servers.

Server **151** accepts a query from user **147** running application **177** and using a search engine (SE) interface. The query may be a general request for data about a certain class of IC chips, for example. The query may contain keywords or a series of keywords describing the desired chips. Alternatively, a phrase may be entered instead of keywords. This depends on any rules that are in place and observed by SW **229**. In normal operation, SW **229** retrieves URLs containing any data matching the user's query as illustrated by the right-angled, double arrow labeled "URLs" placed between server **151** and data store **155**. Data store **155** contains indexed URLs that may contain data that matches a user query.

In this example, such URLs are, as would be the normal case, returned to user premise **147** over the network connection where they appear in a displayed search page within browser window **177**. A user may then select a return link to navigate to the electronic page indexed by the URL link.

Some of the URLs indexed in data store **155** may contain embedded search functions representing private search capabilities along with data matching the criteria of the original query. Those URLs may be automatically assigned for proxy browsing on behalf of the user wherein control **195** and extension **186** are employed to navigate to the pages on behalf of a user and invoke the secondary search engines to return deeper level data or URLs according to the original query. In this case, the interface to auto-log-in function **197** would not be required and no site-logic scripts are used. However, all of the other described modules of FIG. **9** may be employed. Many URLs having private search functions embedded therein may be found during the initial search. Therefore, there may be a rule administered that limits the number of private search engines that may be invoked on behalf of a user. An example of such a rule may be "navigate to only the top ten URLs that match the query by ranking percentage and invoke deeper level searches according to the original query".

In another embodiment, URLs found to contain private search functions are sent back to user premise **147** along with other matching URLs or "hits" and appear in browser **177**, but are listed separately. In this case, a user may select a number of those URLs (containing search functions) for proxy navigation, search execution, and data return. Returned data may, in some cases be delivered as text instead of additional links for manual navigation. In this case, the process would contain an extra step of a user selecting a number of returned URLs containing search functions, and then submitting the selection to SP server **151** for proxy navigation, data search, and data return. Selection may, in some cases, be facilitated by check boxes presented next to each URL. Checking a box indicates to include this URL in proxy navigation.

On-line database **187**, as previously described in FIG. **7**, represents a repository or repositories held by individual WEB sites such as sites **159** and **151** of FIG. **7**. Data and URL links contained in database **187** represent deeper-level WEB site data available through a private search function or through manual link activation and navigation from a main URL. The method and apparatus of the present invention provides a convenient method for searching and returning data held on a deeper level of WEB site depth without requiring a user to manually navigate to the data from a "jump-off page" or, without manually invoking a private search function and entering an additional query to search for the data.

In the embodiment presented herein, it is noted that the exact parameters pertaining to rules for entering queries into private search functions is not pre-known as the system described in this embodiment is not personalized to a user. Therefore, re-structuring of an original query may not be possible. However, it is assumed that some standardization exists with respect to the code used to embed the private search functions as well as with the rules administered for dialog entry into those functions. SW **229** may be pre-programmed then to understand and recognize such standard parameters such that recognition of code and restructuring of a query is still possible. In this instance, known codes and rule-sets would be pre-loaded into a database accessible to SW **229** such that the correct codes and rule-sets may be found by parsing and comparison.

It will be apparent to one with skill in the art that SW **229** may be adapted to work in conjunction with a navigation system in a multi-tasking environment without departing from the spirit and scope of the present invention. For example, many user queries may be processed simultaneously and the only limit to the number of URLs that may be navigated to on behalf of a plurality of users is the processing power of the dedicated node or nodes performing the navigation and data-return functions. In another embodiment, SW **229** and a navigation system may be one application running on one powerful server. Scalability and component distribution may be implemented according to need. There are many possibilities.

The method and apparatus of the present invention may be practiced via private individuals on the Internet, businesses operating on a WAN connected to the Internet, businesses operating via private WAN, and so on.

There are many customizable situations.

The present invention as taught herein and above should be afforded the broadest of scope. The spirit and scope of the present invention is limited only by the claims that follow.

What is claimed is:

**1**. A method for extending an on-line Internet search beyond pre-referenced sources, comprising steps of:

(a) entering a first search criteria in a first search function;

(b) initiating the first search function;

(c) returning in the first search function a pre-referenced first document having data associated with the first search criteria;

(d) testing the first document for an embedded second search function;

(e) on finding a second search function in the first document, automatically entering at least a form of the first search criteria in the second search function; and

(f) returning addresses in the first search function for documents found through the second search function.

**2**. The method of claim **1** wherein the first search function allows natural language in entering search criteria, and further comprises a parsing step for parsing criteria input for significant words and phrases for criteria matches.

**3**. The method of claim **2** wherein the first search function, in step (e), tests the second search function for criteria rules, and amends the first search criteria to conform to the criteria rules.

**4**. The method of claim **1** wherein the first search function is provided by a subscription portal service, and is operated by proxy by subscribers.

**5**. The method of claim **4** wherein the first search function is limited in step (c) to returning first documents pre-registered to a specific subscriber invoking the first search function.

6. An Internet search application comprising:

a first search module having a first criteria interface for entry of a first search criteria;

an inspection function for identifying a second search module in a returned electronic document, the second search module having a second criteria interface; and

an entry module for entering the first search criteria into the search criteria interface of the second search module;

characterized in that the search application, upon entry of a first search criteria in the first criteria interface, returns at least one electronic document having a match to the first search criteria, inspects the document for the second search module, and transfers at least a form of the first search criteria into the second criteria interface.

7. The Internet search application of claim 6 wherein the Internet search application further initiates the second search module after transfer of search criteria, and returns at least addresses of documents found by the second search function in the first search function.

8. The search function of claim 6 wherein the first search module allows natural language criteria entry, and parses entries for significant words and phrases for matching to content in electronic documents returned.

9. The search function of claim 8 wherein the first search module, in step (e), tests the second search module for criteria rules, and amends the first search criteria to conform to the criteria rules.

10. The search function of claim 6 wherein the first search module is provided by a subscription portal service, an is operated by proxy by subscribers.

11. The search function of claim 10 wherein the first search module is limited to returning first documents pre-registered to a specific subscriber invoking the first search function.

* * * * *

US006513031B1

(12) **United States Patent**
**Fries et al.**

(10) **Patent No.: US 6,513,031 B1**
(45) **Date of Patent: Jan. 28, 2003**

(54) **SYSTEM FOR IMPROVING SEARCH AREA SELECTION**

(75) Inventors: **Karen E. Fries**, Seattle, WA (US); **John M. Tippett**, Seattle, WA (US); **Jeffrey Richter**, Bellevue, WA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

( * ) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/219,271**

(22) Filed: **Dec. 23, 1998**

(51) **Int. Cl.**[7] ..... **G06F 17/30**
(52) **U.S. Cl.** ..... **707/3**; 707/5
(58) **Field of Search** ..... 707/1, 3, 5, 4, 707/200

(56) **References Cited**

U.S. PATENT DOCUMENTS

| | | | | |
|---|---|---|---|---|
| 5,175,814 | A * | 12/1992 | Anick et al. | 345/348 |
| 5,467,448 | A | 11/1995 | Hilton et al. | 395/148 |
| 5,696,962 | A * | 12/1997 | Kuplec | 707/4 |
| 5,721,897 | A * | 2/1998 | Rubinstein | 707/3 |
| 5,737,592 | A * | 4/1998 | Nguyen et al. | 707/4 |
| 5,778,363 | A | 7/1998 | Light | 707/5 |
| 5,818,462 | A | 10/1998 | Marks et al. | 345/473 |
| 5,822,731 | A | 10/1998 | Schultz | 704/270.1 |
| 5,864,846 | A * | 1/1999 | Vorhees et al. | 707/5 |
| 5,913,205 | A | 6/1999 | Jain et al. | 707/2 |
| 5,974,412 | A * | 10/1999 | Hazelhurst et al. | 707/3 |
| 5,974,413 | A | 10/1999 | Beauregard et al. | 707/6 |
| 5,987,446 | A * | 11/1999 | Corey et al. | 707/3 |
| 5,987,454 | A * | 11/1999 | Hobbs | 707/4 |
| 6,009,410 | A | 12/1999 | LeMole et al. | 705/14 |
| 6,070,157 | A * | 5/2000 | Jaconsen et al. | 707/1 |
| 6,078,914 | A * | 6/2000 | Redfern | 707/3 |
| 6,199,061 | B1 | 3/2001 | Blewett et al. | 707/3 |
| 6,236,987 | B1 | 5/2001 | Horowitz et al. | 707/3 |
| 6,247,021 | B1 | 6/2001 | Himmel et al. | 707/104 |
| 6,279,017 | B1 | 8/2001 | Walker | 707/529 |

OTHER PUBLICATIONS

Internet Site, "http://www.altavista.digital.com/cgi-bin/query?pg=q&what=web&q=jojo", Mar. 12, 1998.
Internet Site, "http://www.cs.colostate.edu/~dreiling/smartform.html", Mar. 12, 1998.
Internet Site, "http://www.cyber411.com/main.htm", Mar. 12, 1998.
Internet Site, "http://www.dogpile.com/", Mar. 12, 1998.
Internet Site, "http://www.excite.com", Mar. 12, 1998.
Internet Site, "http://www.highway61.com/yak.html", Mar. 12, 1998.
Internet Site, "http://www.hotbot.com/", Mar. 12, 1998.
Internet Site, "http://www.hotbot.com/?MT=&SM=MC&DV=7&RG=.com&DC=10&DE&Ops=MDRTP&_v=&DU=day", Mar. 12, 1998.
Internet Site, "http://www.infoseek.com/", Mar. 12, 1998.
Internet Site, "http://www.infoseek.com/Help?pg=HomeHelp.html", Mar. 12, 1998.
Internet Site, "http://www.isurf.yahoo.com/", Mar. 12, 1998.
Internet Site, "http://www.kresch.com/oss/oss.htm", Mar. 12, 1998.

(List continued on next page.)

*Primary Examiner*—Diane D. Mizrahi
(74) *Attorney, Agent, or Firm*—Theodore M. Magee; Westman, Champlin & Kelly, P.A.

(57) **ABSTRACT**

A method of aiding a user in searching a computer environment includes retrieving a search query from a user, accessing a user profile and selecting a search area based on the search query and the user profile. In other embodiments of the present invention, a method provides for receiving a search query, categorizing at least one term in the search query based on an indexed list of terms derived from pages on a network, and identifying at least one search area based on the category of a search term.

**5 Claims, 30 Drawing Sheets**

I want information on skiing and snowmobiling in Wyoming.

916

OTHER PUBLICATIONS

Internet Site, "http://kresch.com/search/searchme.htm", Mar. 12, 1998.
Internet Site, "http://www.lycos.com/", Mar. 12, 1998.
Internet Site, "http://www.mamma.com/whyuse.html", Mar. 12, 1998.
Internet Site, "http://www.primecomputing.com/ps-search.htm", Mar. 12, 1998.
Internet Site, "http://www.webcrawler.com/", Mar. 12, 1998.
Internet Site, "http://www.yahoo.com/", Mar. 12, 1998.
Internet Site, "http://www.yahoo.com/Computers_and_Internet/Internet/World_Wide_Web/Searching_the Webs/All_in_One_Search", Mar. 12, 1998.
U.S. patent application Ser. No. 09/221,028, Fries, filed Dec. 23, 1998.
U.S. patent application Ser. No. 09/221,659, Fries et al., filed Dec. 23, 1998.
U.S. patent application Ser. No. 09/221,663, Fries et al., filed Dec. 23, 1998.

* cited by examiner

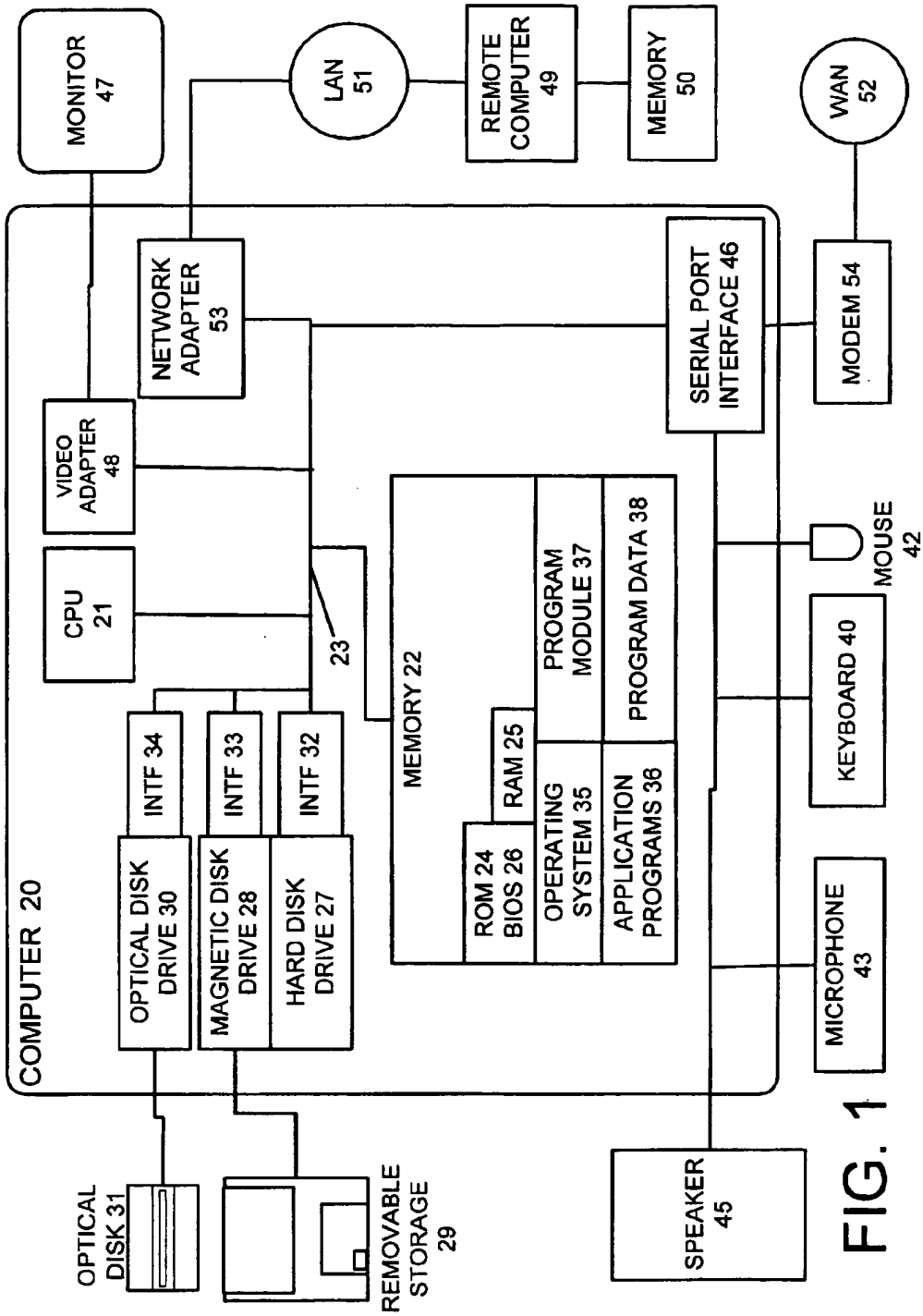

MONITOR 47
LAN 51
REMOTE COMPUTER 49
MEMORY 50
WAN 52
NETWORK ADAPTER 53
SERIAL PORT INTERFACE 46
MODEM 54
VIDEO ADAPTER 48
CPU 21
23
MEMORY 22
PROGRAM MODULE 37
PROGRAM DATA 38
MOUSE 42
RAM 25
ROM 24 BIOS 26
OPERATING SYSTEM 35
APPLICATION PROGRAMS 36
KEYBOARD 40
INTF 34
INTF 33
INTF 32
COMPUTER 20
OPTICAL DISK DRIVE 30
MAGNETIC DISK DRIVE 28
HARD DISK DRIVE 27
MICROPHONE 43
OPTICAL DISK 31
REMOVABLE STORAGE 29
SPEAKER 45

FIG. 1

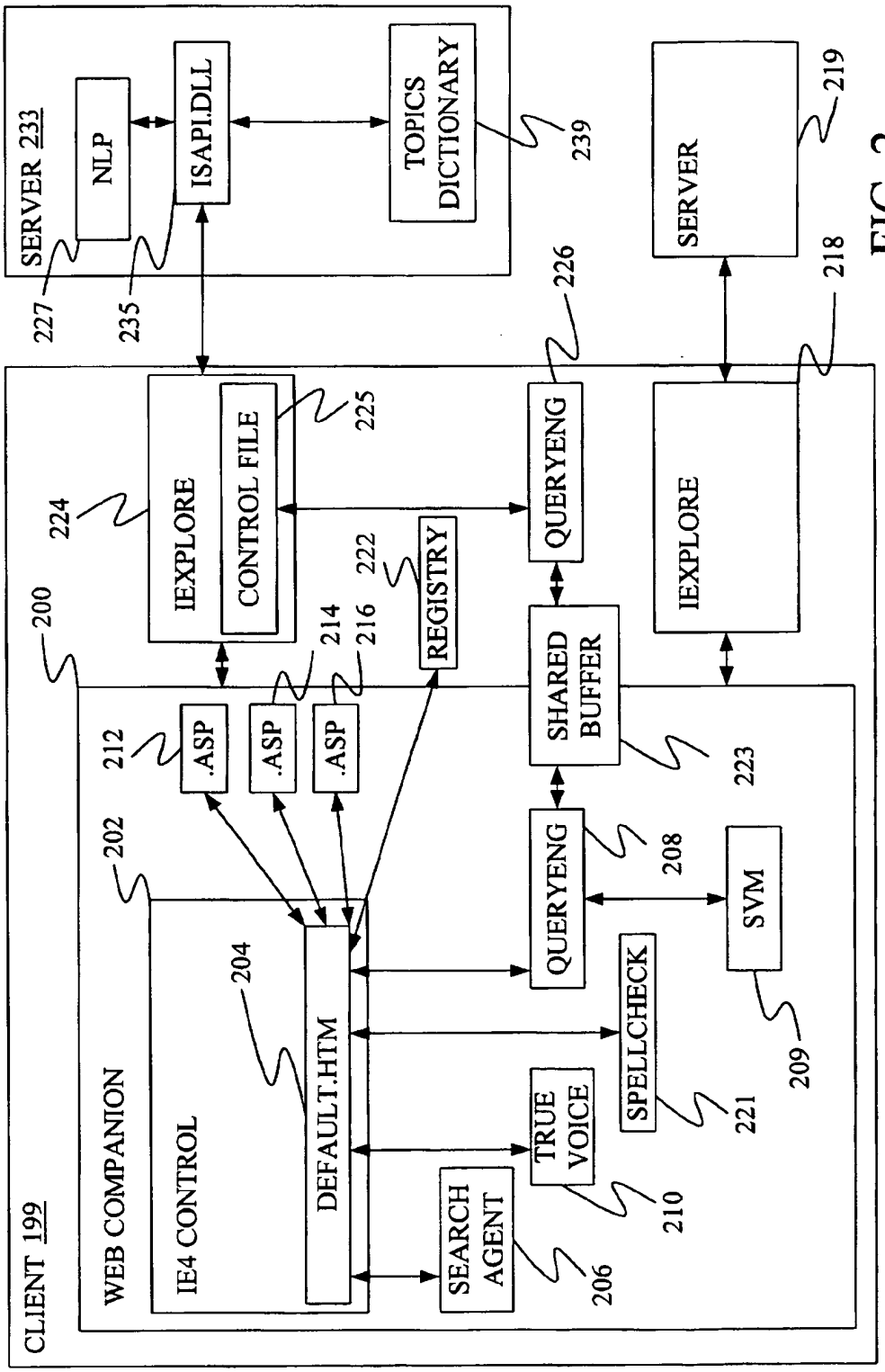

SERVER 233
NLP
ISAPI.DLL
TOPICS DICTIONARY
239
227
235
SERVER
219
218
226
CLIENT 199
WEB COMPANION
IE4 CONTROL
IEXPLORE
CONTROL FILE
224
225
QUERYENG
IEXPLORE
200
REGISTRY
222
214
216
SHARED BUFFER
223
.ASP
.ASP
.ASP
212
202
204
DEFAULT.HTM
QUERYENG
208
SVM
209
SPELLCHECK
221
TRUE VOICE
210
SEARCH AGENT
206

FIG. 2

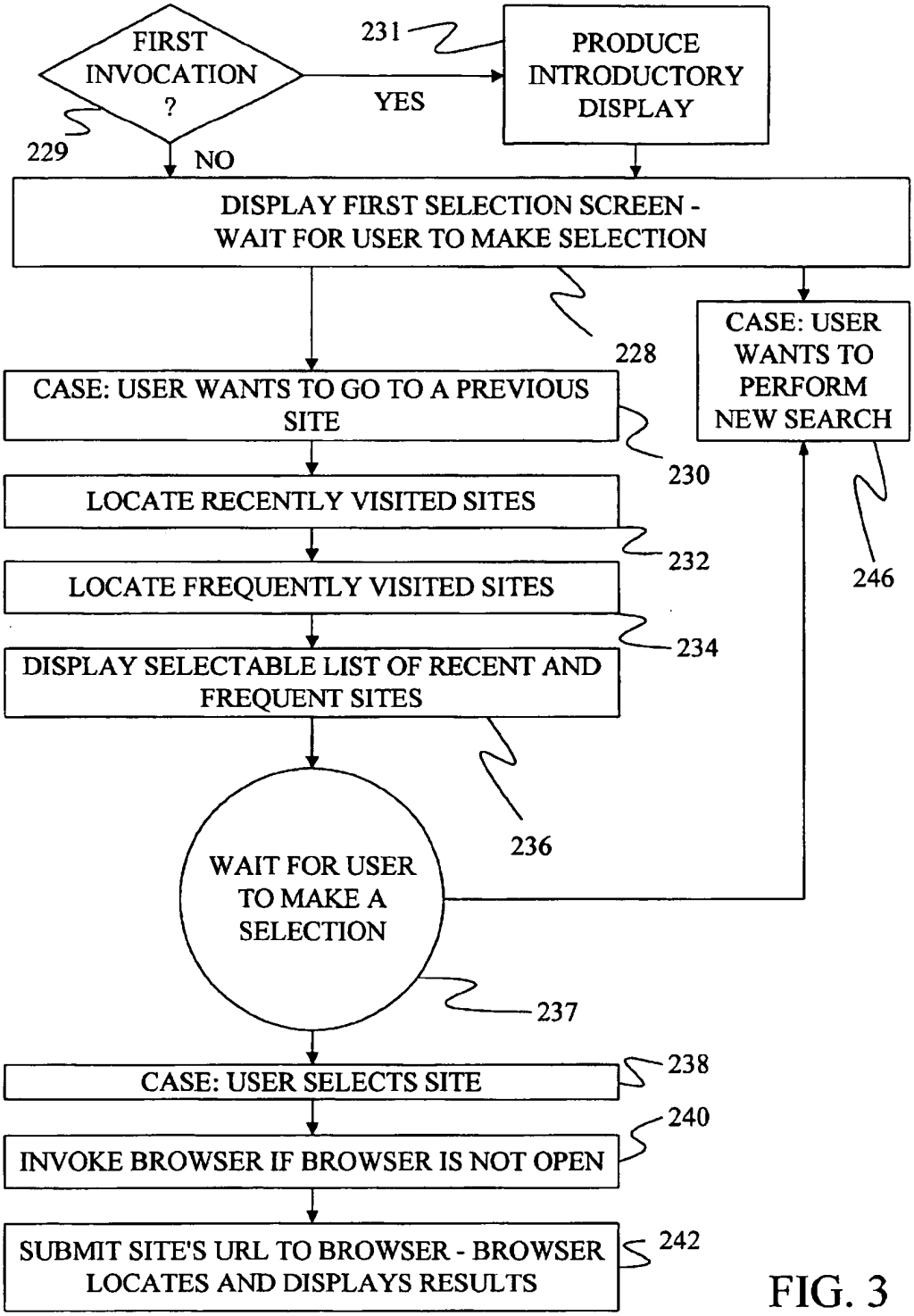

FIRST INVOCATION ?
231
YES
PRODUCE INTRODUCTORY DISPLAY
229
NO
DISPLAY FIRST SELECTION SCREEN - WAIT FOR USER TO MAKE SELECTION
228
CASE: USER WANTS TO GO TO A PREVIOUS SITE
230
CASE: USER WANTS TO PERFORM NEW SEARCH
246
LOCATE RECENTLY VISITED SITES
232
LOCATE FREQUENTLY VISITED SITES
234
DISPLAY SELECTABLE LIST OF RECENT AND FREQUENT SITES
236
WAIT FOR USER TO MAKE A SELECTION
237
CASE: USER SELECTS SITE
238
INVOKE BROWSER IF BROWSER IS NOT OPEN
240
SUBMIT SITE'S URL TO BROWSER - BROWSER LOCATES AND DISPLAYS RESULTS
242

FIG. 3

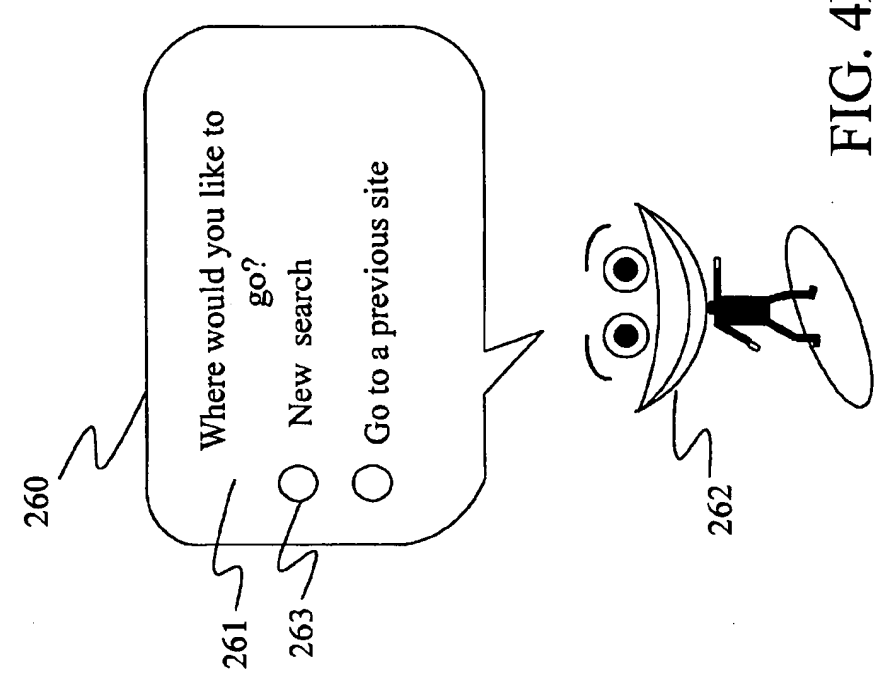

260
Where would you like to go?
New search
Go to a previous site
261
263
262

FIG. 4B

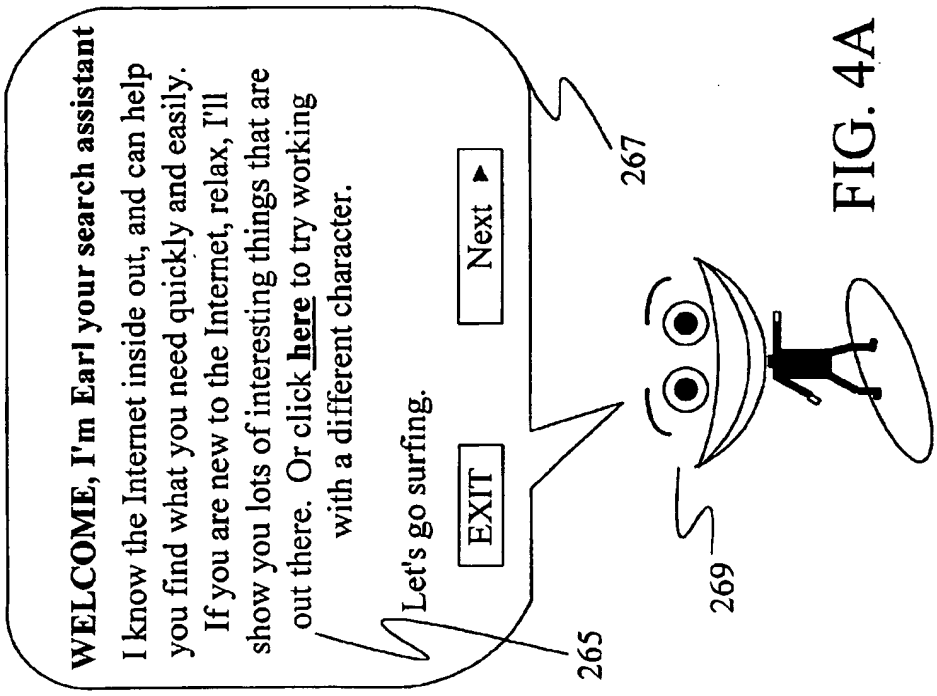

WELCOME, I'm Earl your search assistant
I know the Internet inside out, and can help you find what you need quickly and easily.
If you are new to the Internet, relax, I'll show you lots of interesting things that are out there. Or click here to try working with a different character.
Let's go surfing.
EXIT
Next
265
267
269

FIG. 4A

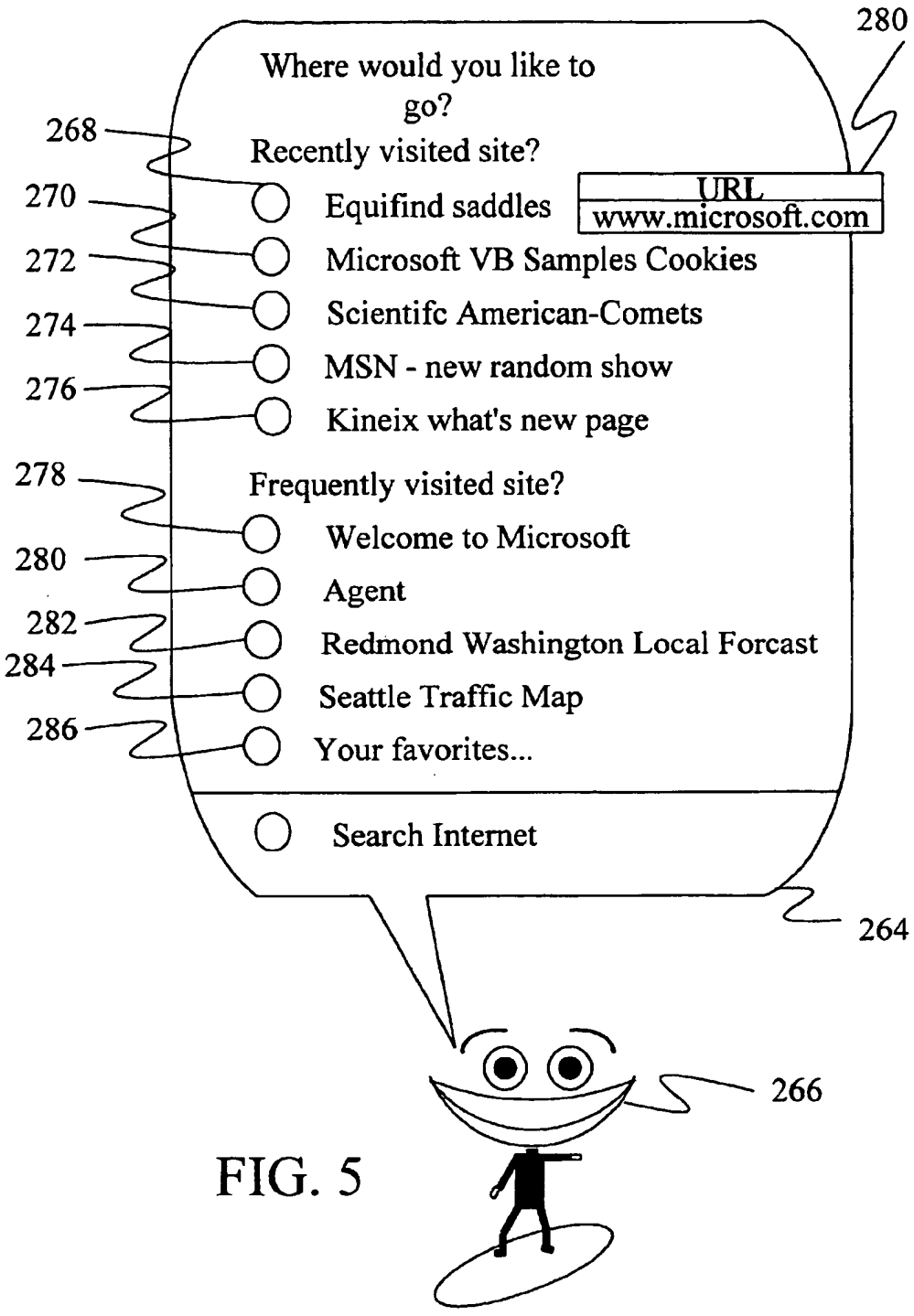

280
Where would you like to go?
268
Recently visited site?
URL
www.microsoft.com
270
Equifind saddles
272
Microsoft VB Samples Cookies
274
Scientifc American-Comets
276
MSN - new random show
Kineix what's new page
278
Frequently visited site?
Welcome to Microsoft
280
Agent
282
Redmond Washington Local Forcast
284
Seattle Traffic Map
286
Your favorites...
Search Internet
264
266

FIG. 5

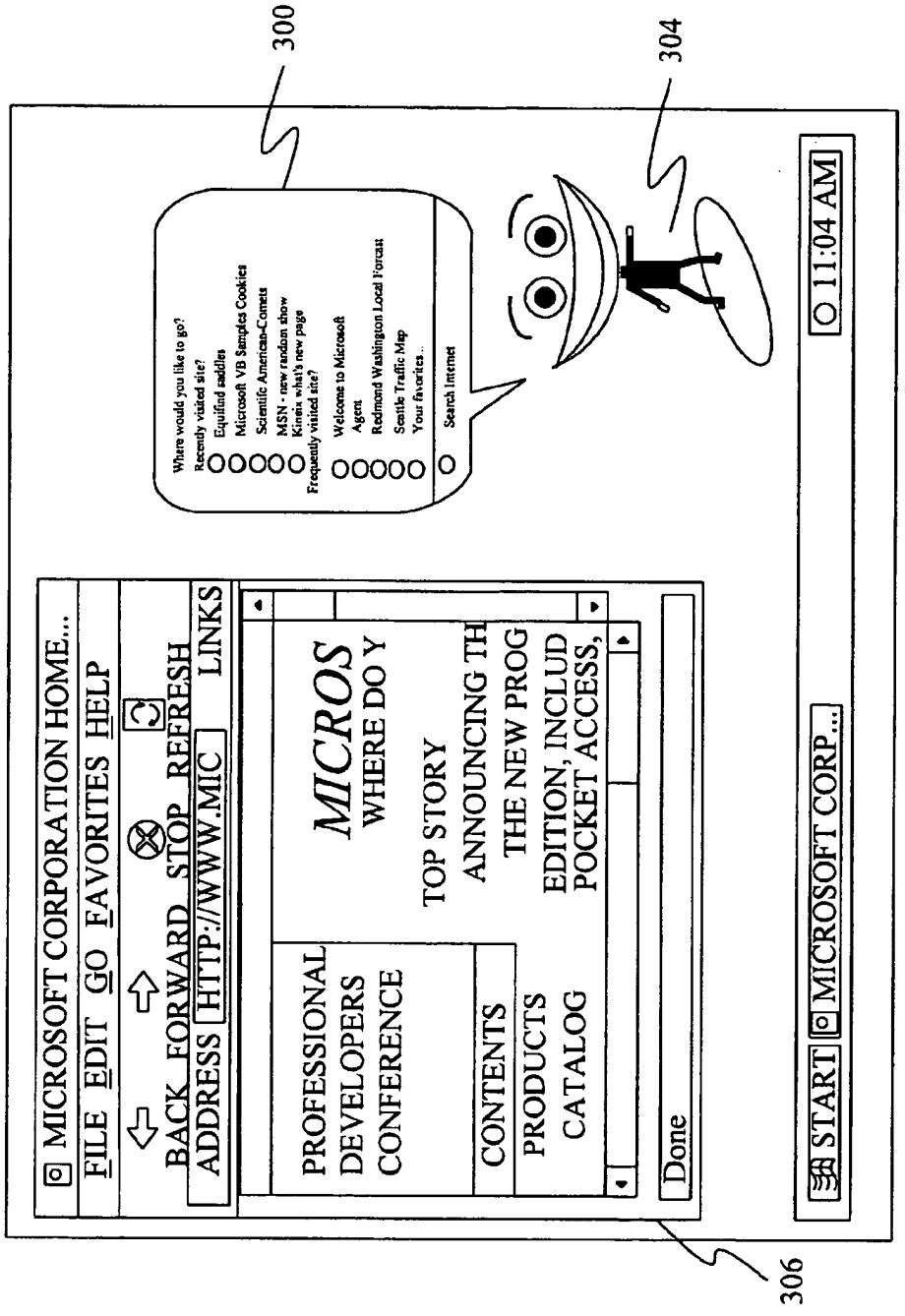

300
304
306
MICROSOFT CORPORATION HOME...
FILE EDIT GO FAVORITES HELP
BACK FORWARD STOP REFRESH
ADDRESS HTTP://WWW.MIC
LINKS
PROFESSIONAL DEVELOPERS CONFERENCE
CONTENTS
PRODUCTS CATALOG
MICROS
WHERE DO Y
TOP STORY
ANNOUNCING TH
THE NEW PROG
EDITION, INCLUD
POCKET ACCESS,
Done
START
MICROSOFT CORP...
11:04 AM
Where would you like to go?
Recently visited site?
Equifind saddles
Microsoft VB Samples Cookies
Scientific American-Comets
MSN - new random show
Kinsis what's new page
Frequently visited site?
Welcome to Microsoft
Agent
Redmond Washington Local Forcast
Seattle Traffic Map
Your favorites...
Search Internet

FIG. 6

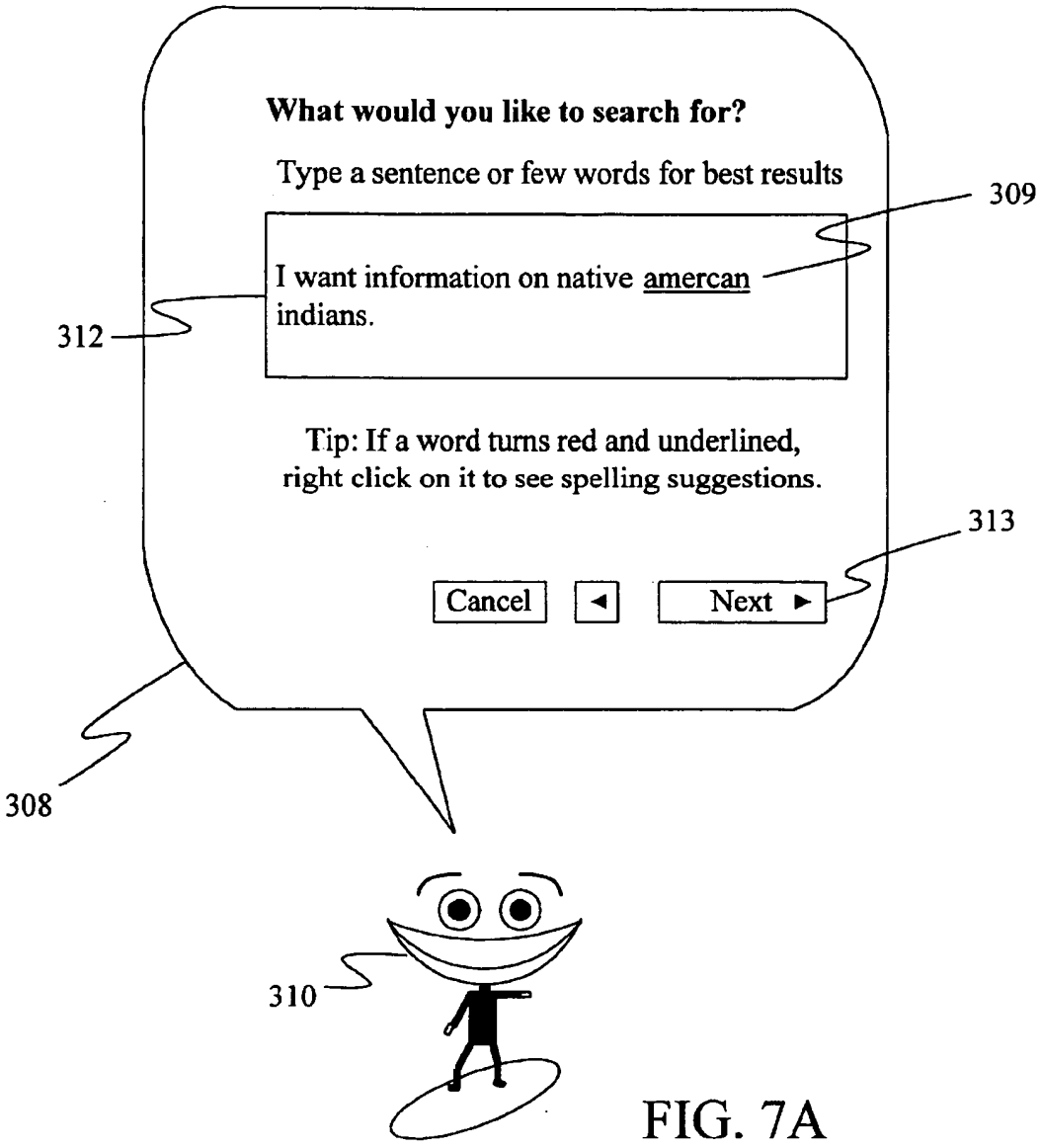

What would you like to search for?
Type a sentence or few words for best results
309
I want information on native amercan indians.
312
Tip: If a word turns red and underlined, right click on it to see spelling suggestions.
313
Cancel
Next
308
310

FIG. 7A

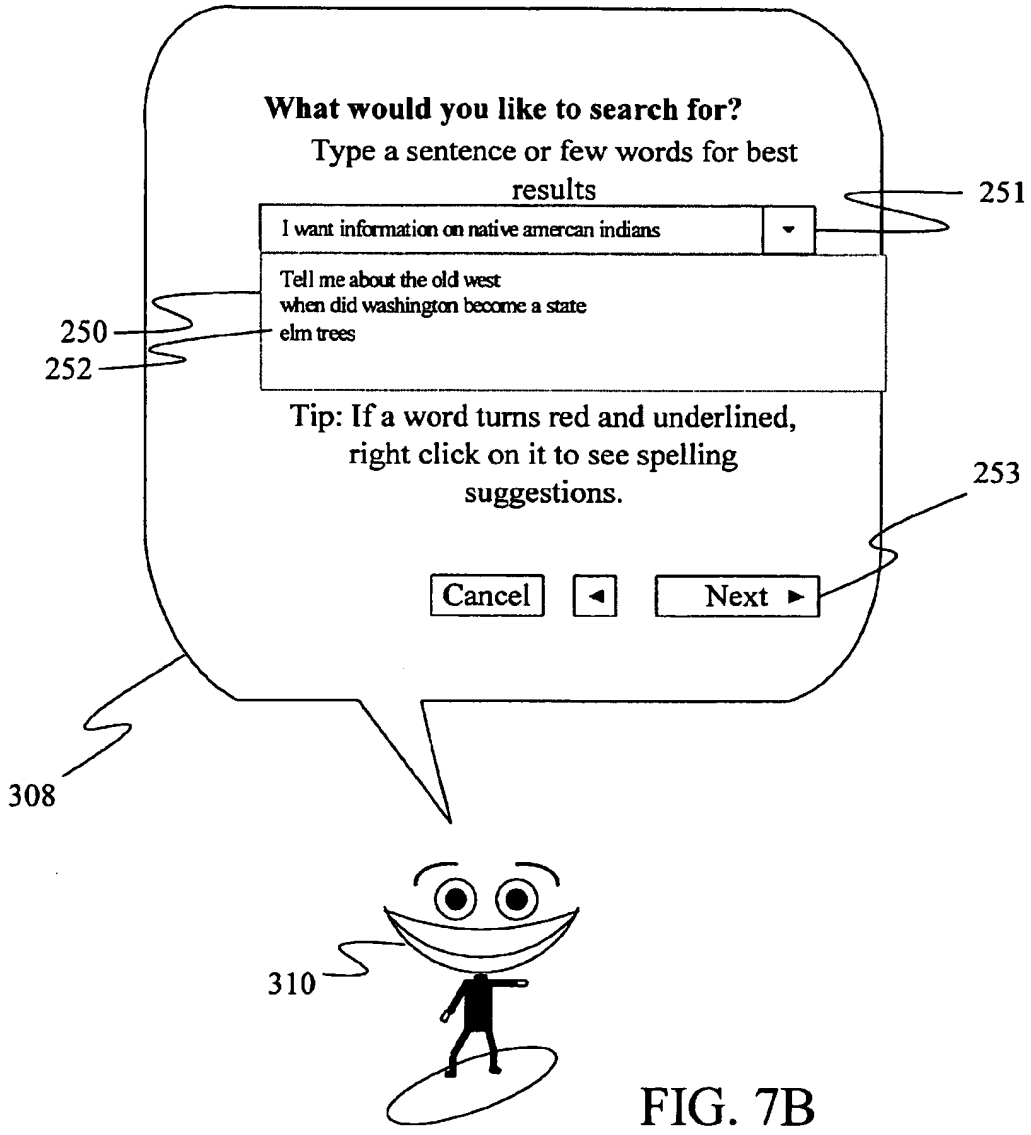

What would you like to search for?
Type a sentence or few words for best results
I want information on native american indians
Tell me about the old west
when did washington become a state
elm trees
Tip: If a word turns red and underlined, right click on it to see spelling suggestions.
Cancel
Next
250
251
252
253
308
310

FIG. 7B

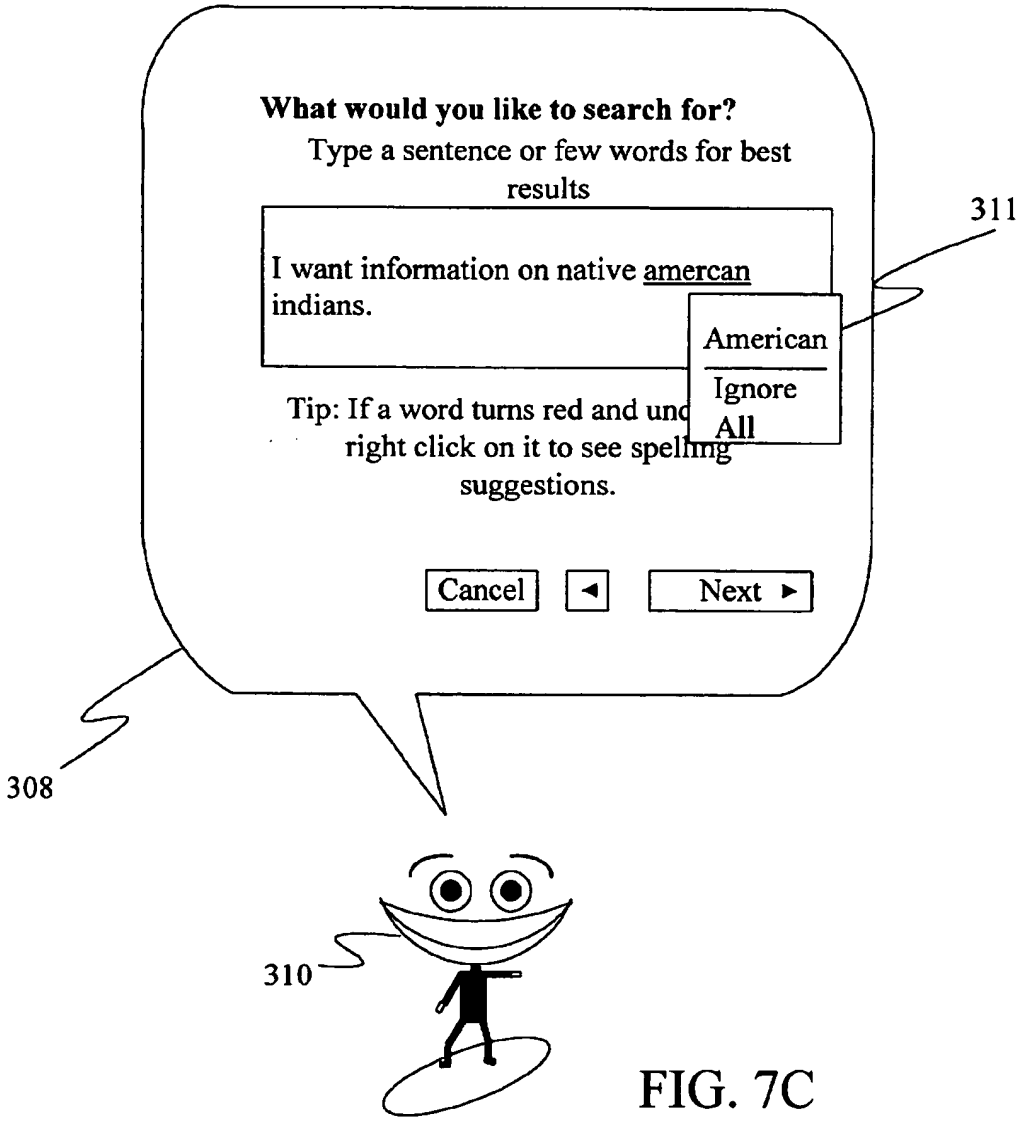

What would you like to search for?
Type a sentence or few words for best results
I want information on native amercan indians.
American
Ignore
All
Tip: If a word turns red and unc
right click on it to see spelling
suggestions.
Cancel
Next
311
308
310

FIG. 7C

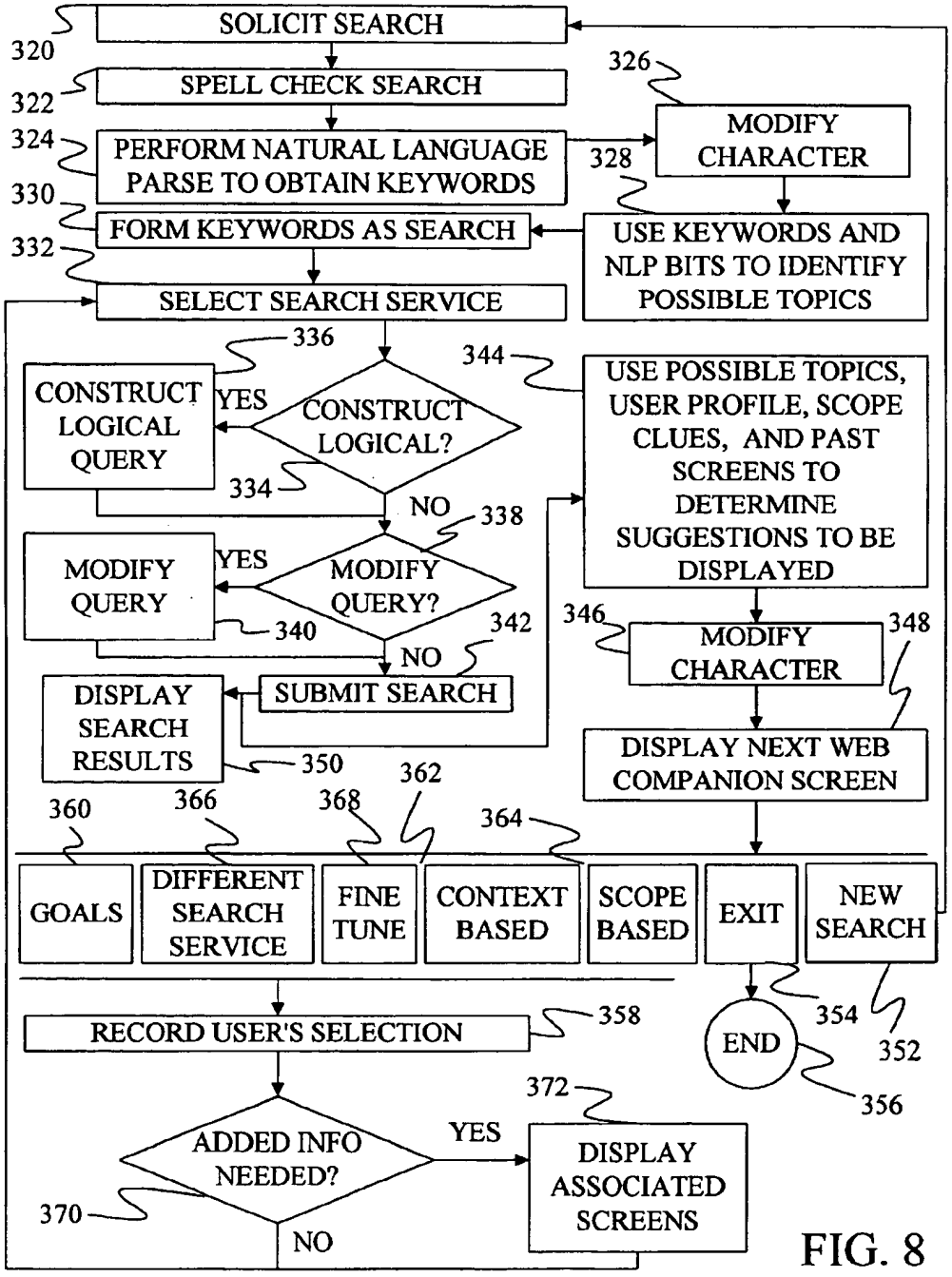

SOLICIT SEARCH
320
SPELL CHECK SEARCH
322
PERFORM NATURAL LANGUAGE PARSE TO OBTAIN KEYWORDS
324
FORM KEYWORDS AS SEARCH
330
SELECT SEARCH SERVICE
332
MODIFY CHARACTER
326
328
USE KEYWORDS AND NLP BITS TO IDENTIFY POSSIBLE TOPICS
336
CONSTRUCT LOGICAL QUERY
YES
CONSTRUCT LOGICAL?
334
NO
338
MODIFY QUERY
YES
MODIFY QUERY?
340
NO
342
SUBMIT SEARCH
DISPLAY SEARCH RESULTS
350
344
USE POSSIBLE TOPICS, USER PROFILE, SCOPE CLUES, AND PAST SCREENS TO DETERMINE SUGGESTIONS TO BE DISPLAYED
346
MODIFY CHARACTER
348
DISPLAY NEXT WEB COMPANION SCREEN
360
366
368
362
364
GOALS
DIFFERENT SEARCH SERVICE
FINE TUNE
CONTEXT BASED
SCOPE BASED
EXIT
NEW SEARCH
RECORD USER'S SELECTION
358
END
354
352
356
372
ADDED INFO NEEDED?
YES
370
NO
DISPLAY ASSOCIATED SCREENS

FIG. 8

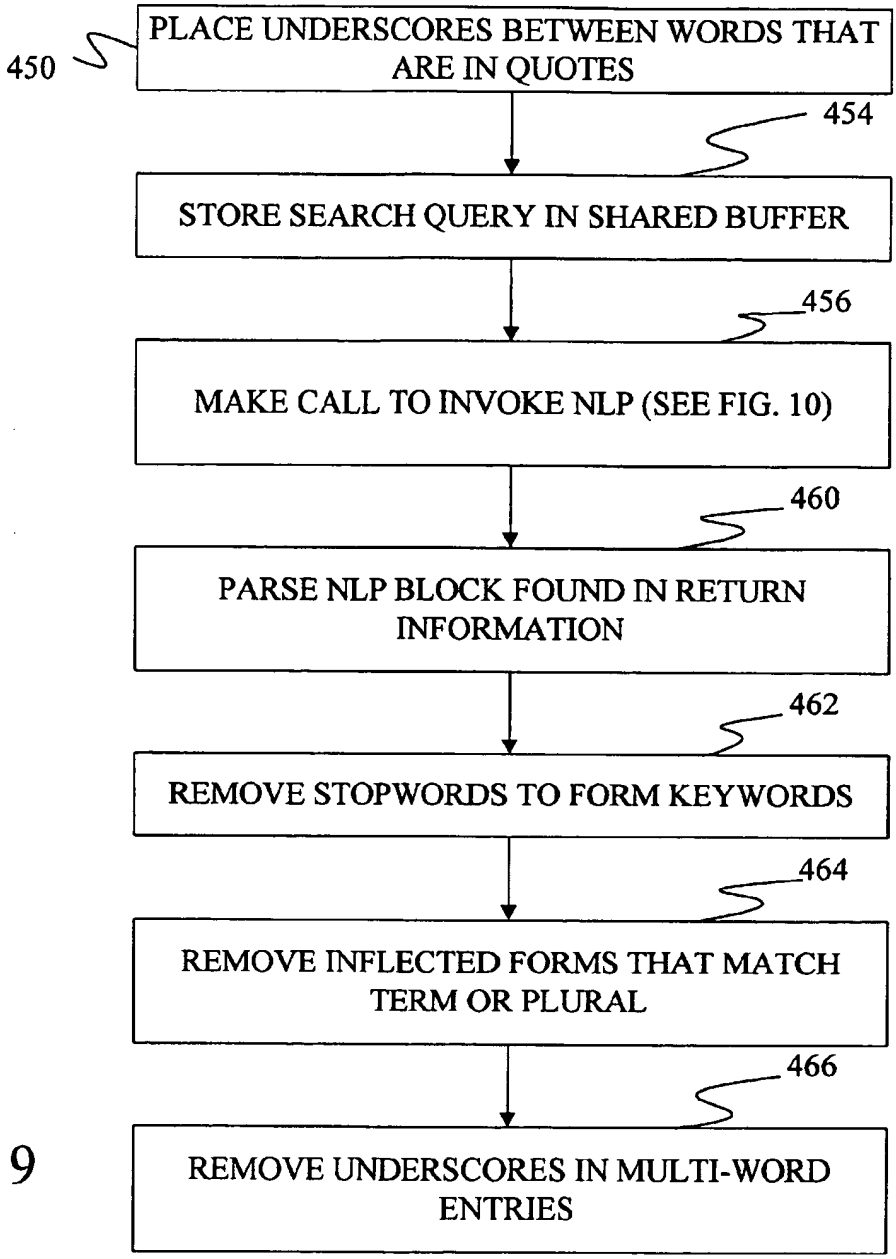

450
PLACE UNDERSCORES BETWEEN WORDS THAT ARE IN QUOTES
454
STORE SEARCH QUERY IN SHARED BUFFER
456
MAKE CALL TO INVOKE NLP (SEE FIG. 10)
460
PARSE NLP BLOCK FOUND IN RETURN INFORMATION
462
REMOVE STOPWORDS TO FORM KEYWORDS
464
REMOVE INFLECTED FORMS THAT MATCH TERM OR PLURAL
466
REMOVE UNDERSCORES IN MULTI-WORD ENTRIES

FIG. 9

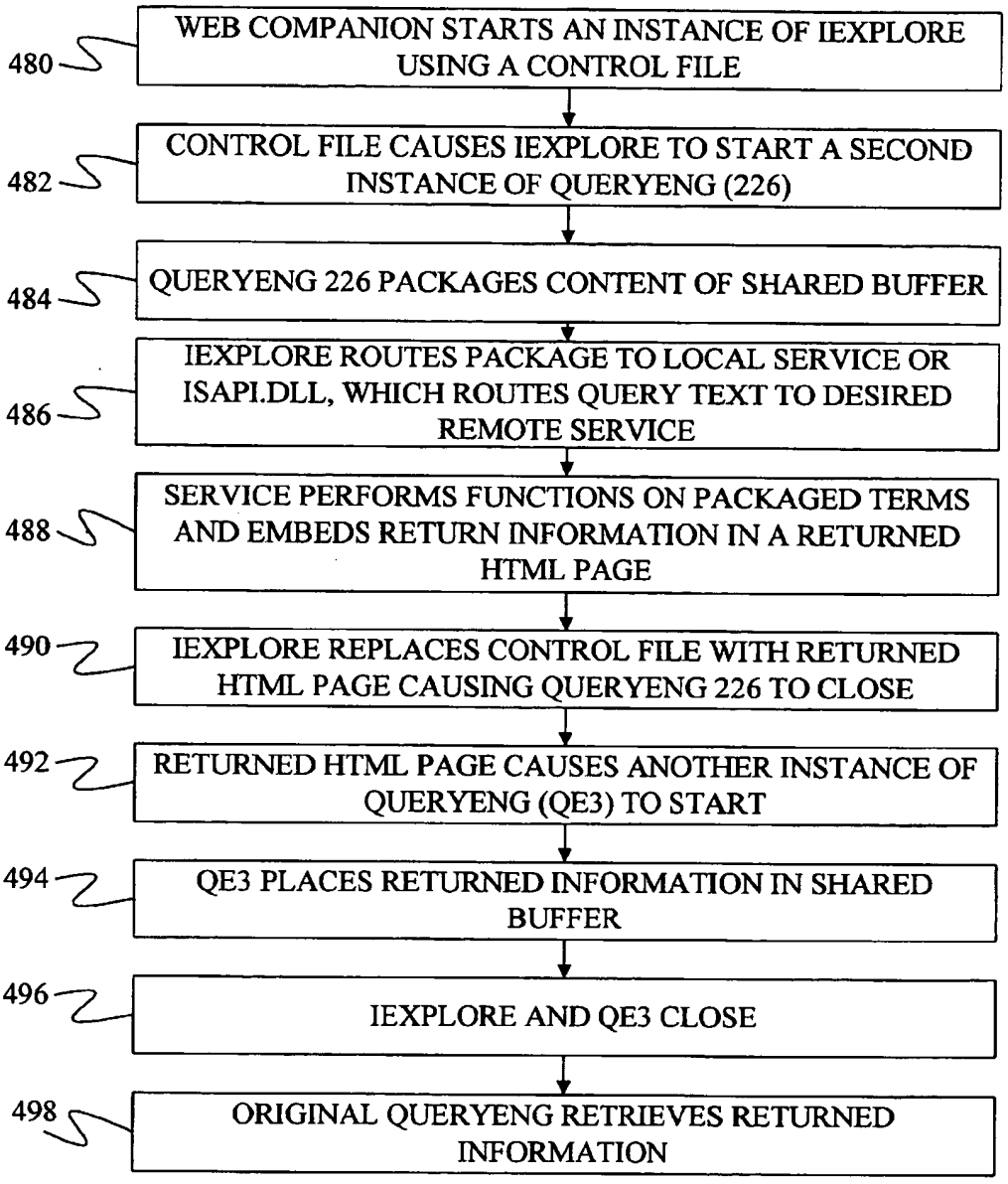

480
WEB COMPANION STARTS AN INSTANCE OF IEXPLORE USING A CONTROL FILE
482
CONTROL FILE CAUSES IEXPLORE TO START A SECOND INSTANCE OF QUERYENG (226)
484
QUERYENG 226 PACKAGES CONTENT OF SHARED BUFFER
486
IEXPLORE ROUTES PACKAGE TO LOCAL SERVICE OR ISAPI.DLL, WHICH ROUTES QUERY TEXT TO DESIRED REMOTE SERVICE
488
SERVICE PERFORMS FUNCTIONS ON PACKAGED TERMS AND EMBEDS RETURN INFORMATION IN A RETURNED HTML PAGE
490
IEXPLORE REPLACES CONTROL FILE WITH RETURNED HTML PAGE CAUSING QUERYENG 226 TO CLOSE
492
RETURNED HTML PAGE CAUSES ANOTHER INSTANCE OF QUERYENG (QE3) TO START
494
QE3 PLACES RETURNED INFORMATION IN SHARED BUFFER
496
IEXPLORE AND QE3 CLOSE
498
ORIGINAL QUERYENG RETRIEVES RETURNED INFORMATION

FIG. 10

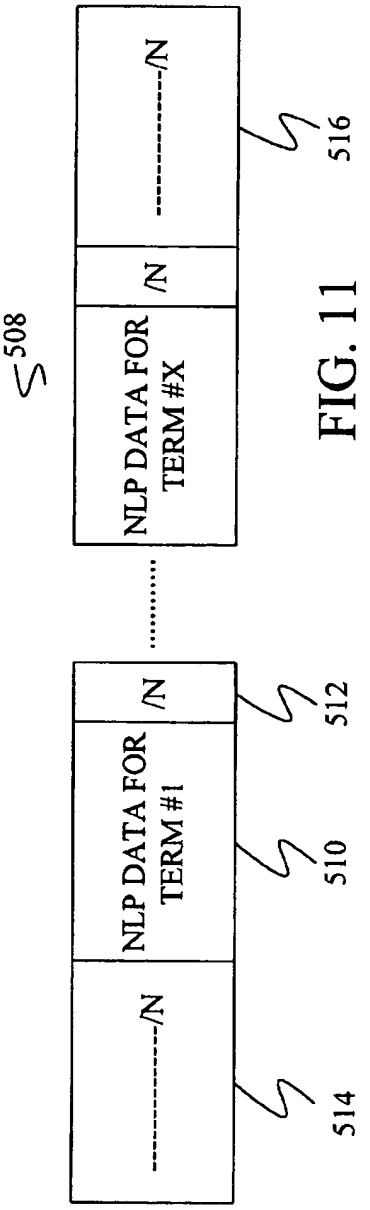

508
NLP DATA FOR TERM #X
/N
----------/N
516
NLP DATA FOR TERM #1
/N
----------/N
510
512
514

FIG. 11

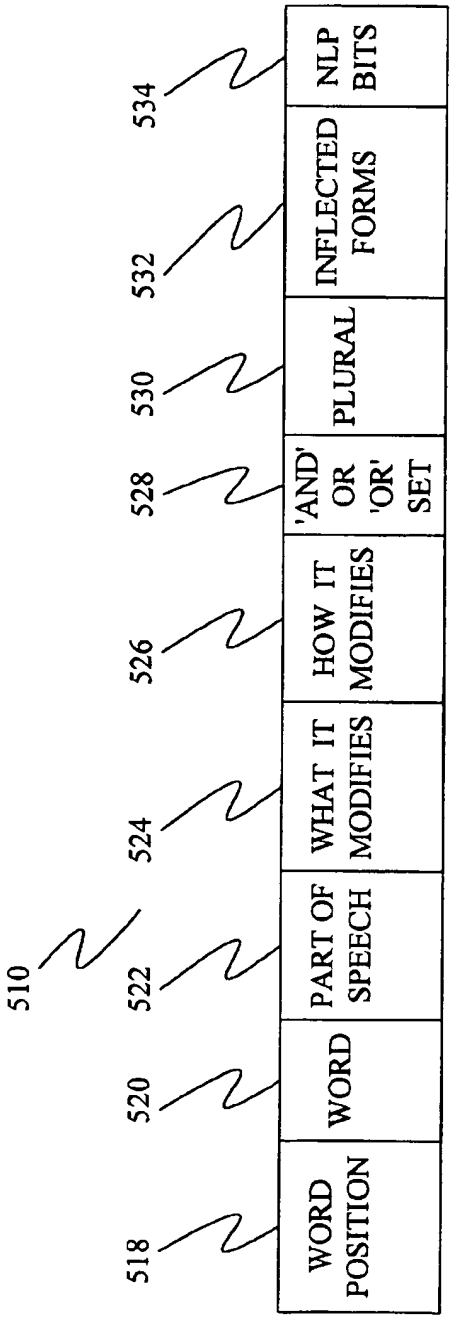

510
WORD POSITION
WORD
PART OF SPEECH
WHAT IT MODIFIES
HOW IT MODIFIES
'AND' OR 'OR' SET
PLURAL
INFLECTED FORMS
NLP BITS
518
520
522
524
526
528
530
532
534

FIG. 12

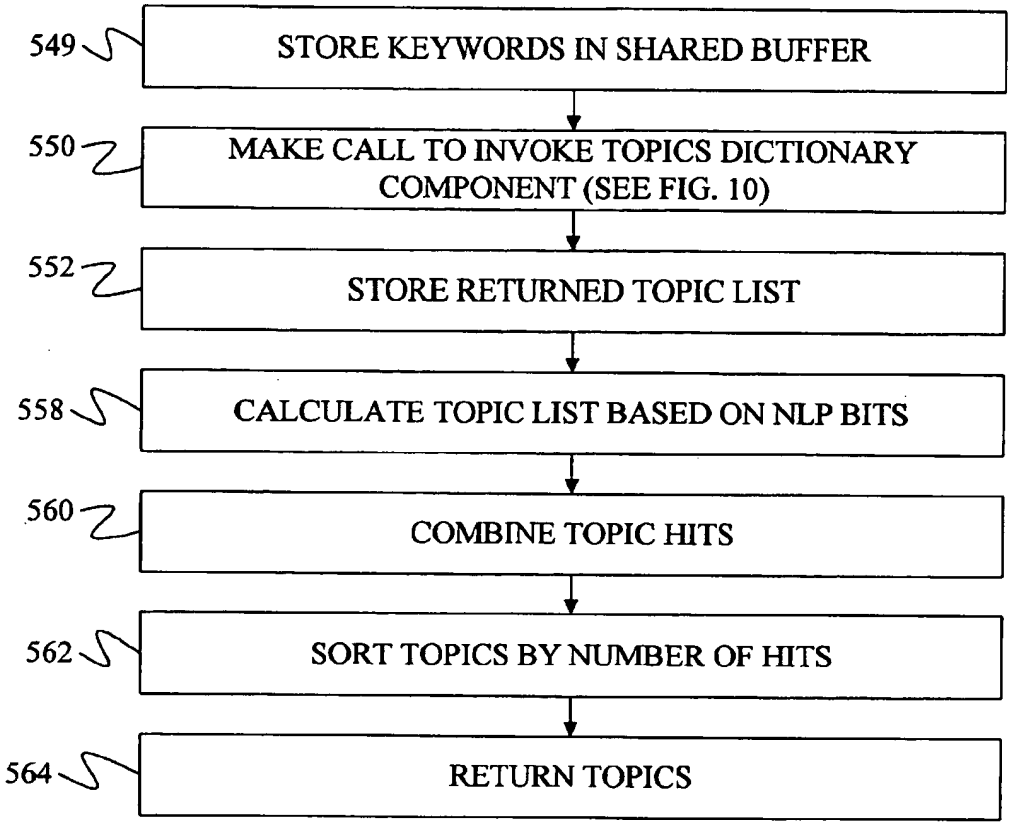

549
STORE KEYWORDS IN SHARED BUFFER
550
MAKE CALL TO INVOKE TOPICS DICTIONARY COMPONENT (SEE FIG. 10)
552
STORE RETURNED TOPIC LIST
558
CALCULATE TOPIC LIST BASED ON NLP BITS
560
COMBINE TOPIC HITS
562
SORT TOPICS BY NUMBER OF HITS
564
RETURN TOPICS

FIG. 13

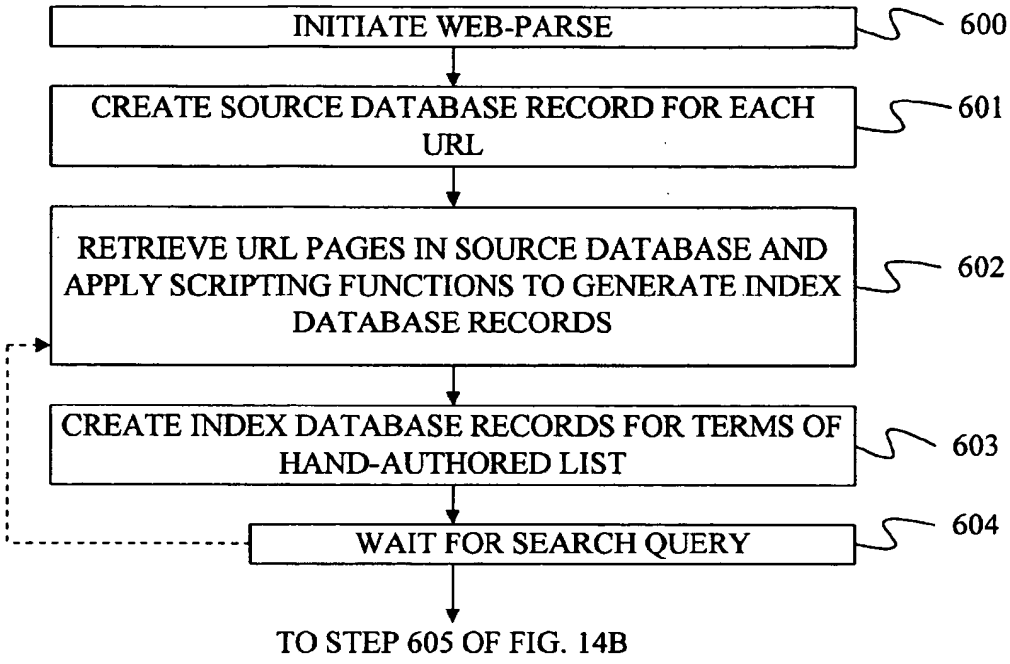

INITIATE WEB-PARSE
600
CREATE SOURCE DATABASE RECORD FOR EACH URL
601
RETRIEVE URL PAGES IN SOURCE DATABASE AND APPLY SCRIPTING FUNCTIONS TO GENERATE INDEX DATABASE RECORDS
602
CREATE INDEX DATABASE RECORDS FOR TERMS OF HAND-AUTHORED LIST
603
WAIT FOR SEARCH QUERY
604
TO STEP 605 OF FIG. 14B

FIG. 14A

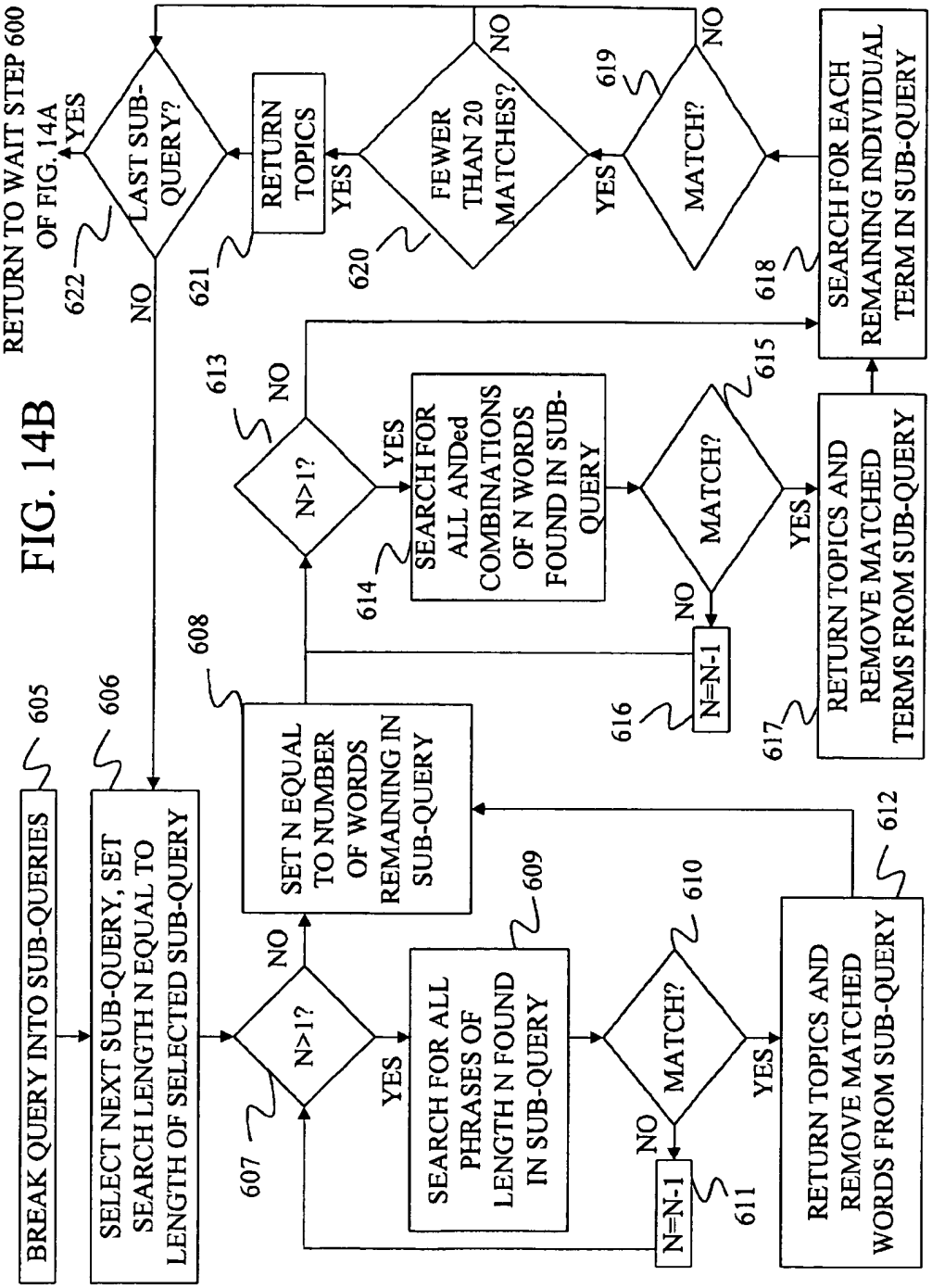

BREAK QUERY INTO SUB-QUERIES
605
SELECT NEXT SUB-QUERY, SET SEARCH LENGTH N EQUAL TO LENGTH OF SELECTED SUB-QUERY
606
N>1?
607
NO
YES
SET N EQUAL TO NUMBER OF WORDS REMAINING IN SUB-QUERY
608
SEARCH FOR ALL PHRASES OF LENGTH N FOUND IN SUB-QUERY
609
MATCH?
610
NO
YES
N=N-1
611
RETURN TOPICS AND REMOVE MATCHED WORDS FROM SUB-QUERY
612
N>1?
613
NO
YES
SEARCH FOR ALL ANDed COMBINATIONS OF N WORDS FOUND IN SUB-QUERY
614
MATCH?
615
NO
YES
N=N-1
616
RETURN TOPICS AND REMOVE MATCHED TERMS FROM SUB-QUERY
617
SEARCH FOR EACH REMAINING INDIVIDUAL TERM IN SUB-QUERY
618
MATCH?
619
NO
YES
FEWER THAN 20 MATCHES?
620
NO
YES
RETURN TOPICS
621
LAST SUB-QUERY?
622
YES
NO
RETURN TO WAIT STEP 600 OF FIG. 14A

FIG. 14B

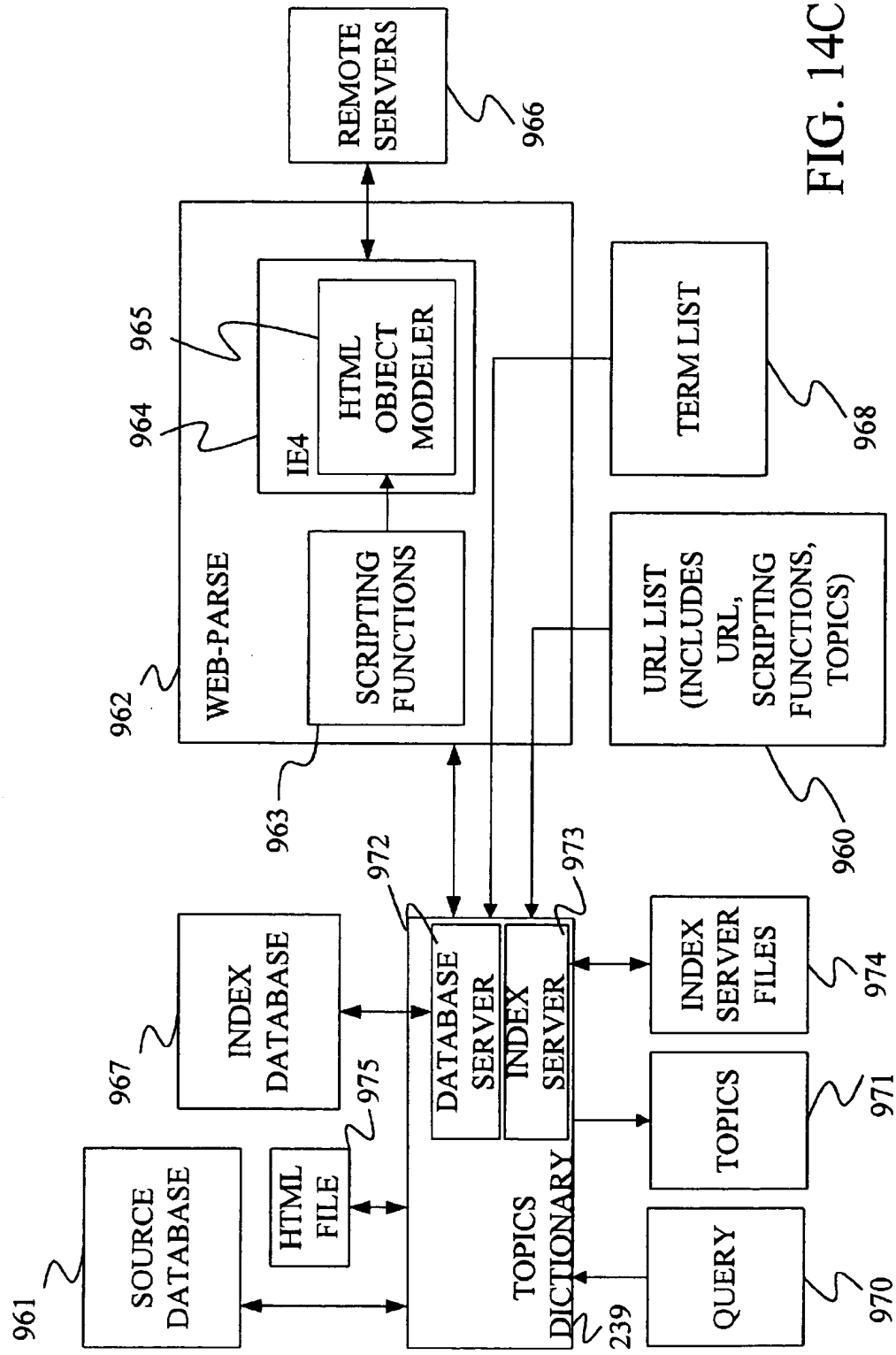

REMOTE SERVERS
966
965
964
IE4
HTML OBJECT MODELER
TERM LIST
968
WEB-PARSE
962
SCRIPTING FUNCTIONS
URL LIST (INCLUDES URL, SCRIPTING FUNCTIONS, TOPICS)
960
963
972
973
INDEX DATABASE
967
DATABASE SERVER
INDEX SERVER
INDEX SERVER FILES
974
975
HTML FILE
TOPICS
971
SOURCE DATABASE
961
TOPICS DICTIONARY
239
QUERY
970

FIG. 14C

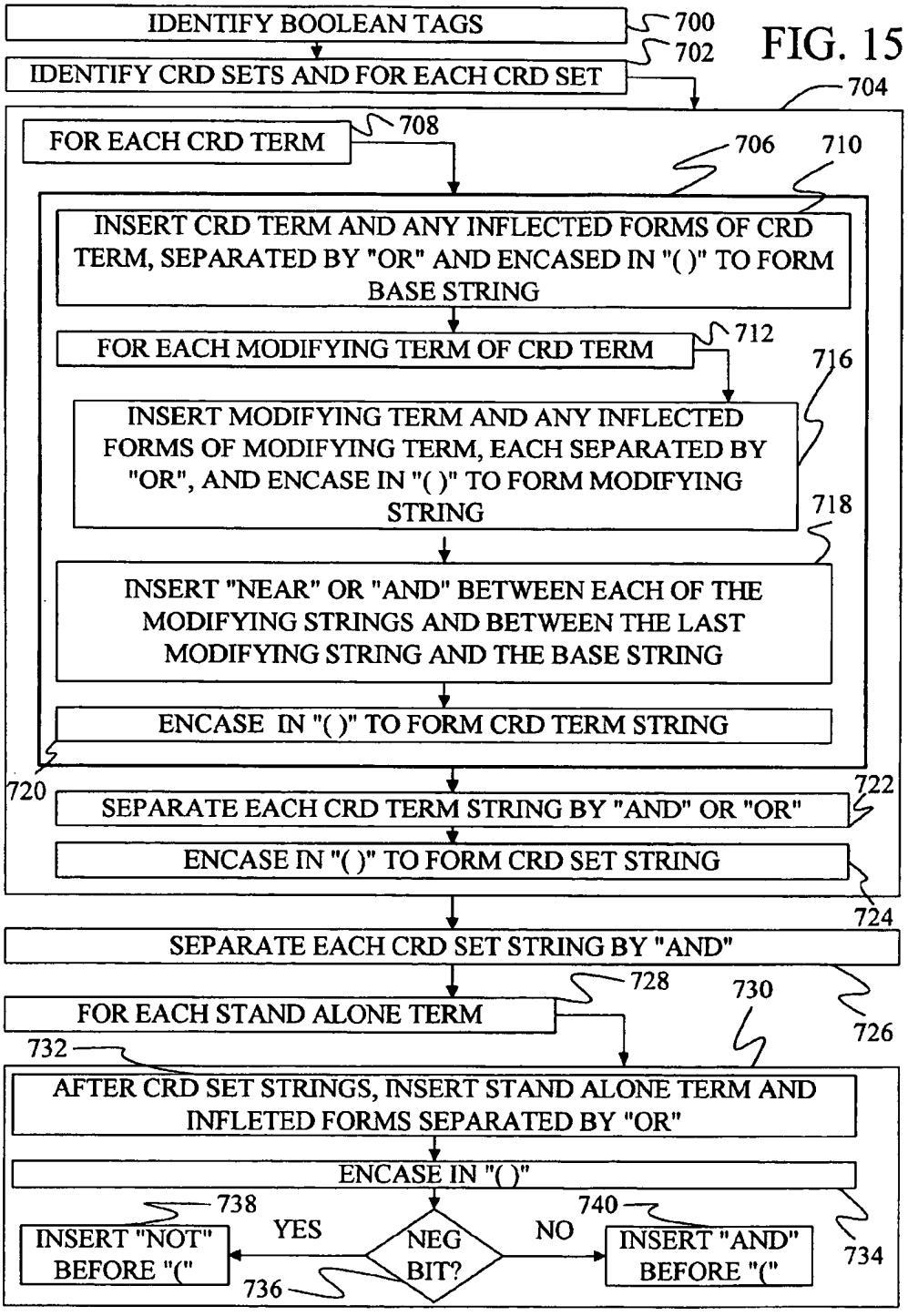

IDENTIFY BOOLEAN TAGS
700
IDENTIFY CRD SETS AND FOR EACH CRD SET
702
704
FOR EACH CRD TERM
708
706
710
INSERT CRD TERM AND ANY INFLECTED FORMS OF CRD TERM, SEPARATED BY "OR" AND ENCASED IN "( )" TO FORM BASE STRING
FOR EACH MODIFYING TERM OF CRD TERM
712
716
INSERT MODIFYING TERM AND ANY INFLECTED FORMS OF MODIFYING TERM, EACH SEPARATED BY "OR", AND ENCASE IN "( )" TO FORM MODIFYING STRING
718
INSERT "NEAR" OR "AND" BETWEEN EACH OF THE MODIFYING STRINGS AND BETWEEN THE LAST MODIFYING STRING AND THE BASE STRING
ENCASE IN "( )" TO FORM CRD TERM STRING
720
SEPARATE EACH CRD TERM STRING BY "AND" OR "OR"
722
ENCASE IN "( )" TO FORM CRD SET STRING
724
SEPARATE EACH CRD SET STRING BY "AND"
728
730
FOR EACH STAND ALONE TERM
726
732
AFTER CRD SET STRINGS, INSERT STAND ALONE TERM AND INFLETED FORMS SEPARATED BY "OR"
ENCASE IN "( )"
738
740
INSERT "NOT" BEFORE "("
YES
NEG BIT?
NO
INSERT "AND" BEFORE "("
736
734

FIG. 15

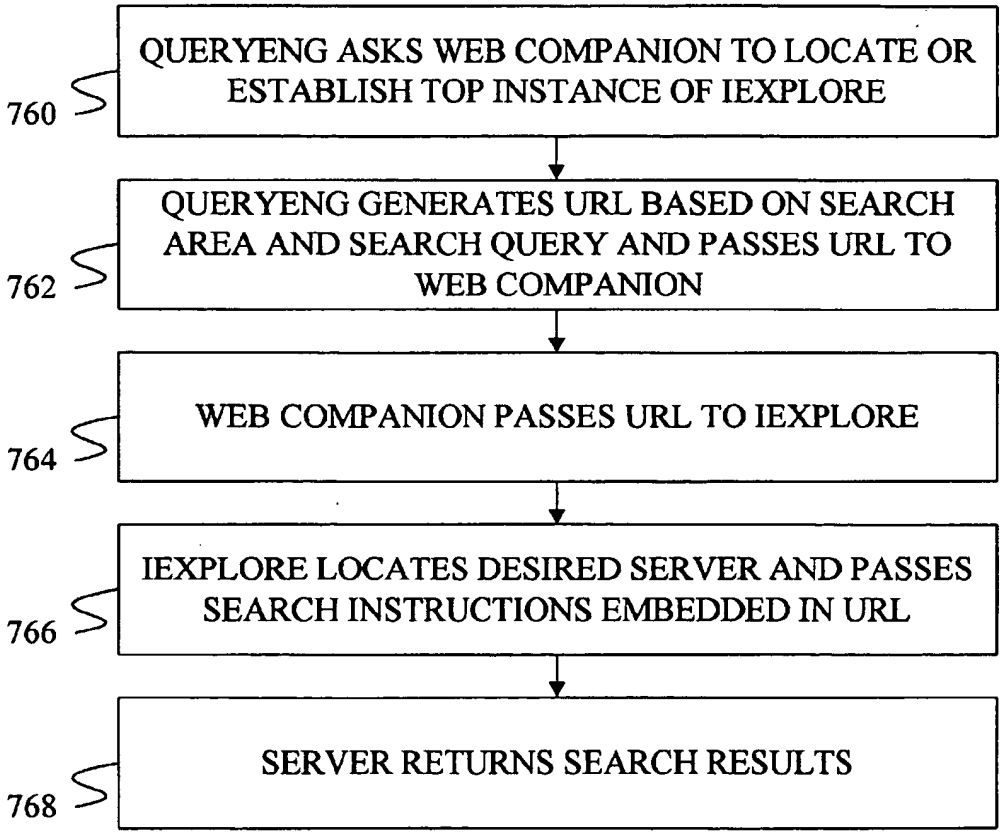

QUERYENG ASKS WEB COMPANION TO LOCATE OR ESTABLISH TOP INSTANCE OF IEXPLORE
760
QUERYENG GENERATES URL BASED ON SEARCH AREA AND SEARCH QUERY AND PASSES URL TO WEB COMPANION
762
WEB COMPANION PASSES URL TO IEXPLORE
764
IEXPLORE LOCATES DESIRED SERVER AND PASSES SEARCH INSTRUCTIONS EMBEDDED IN URL
766
SERVER RETURNS SEARCH RESULTS
768

FIG. 16

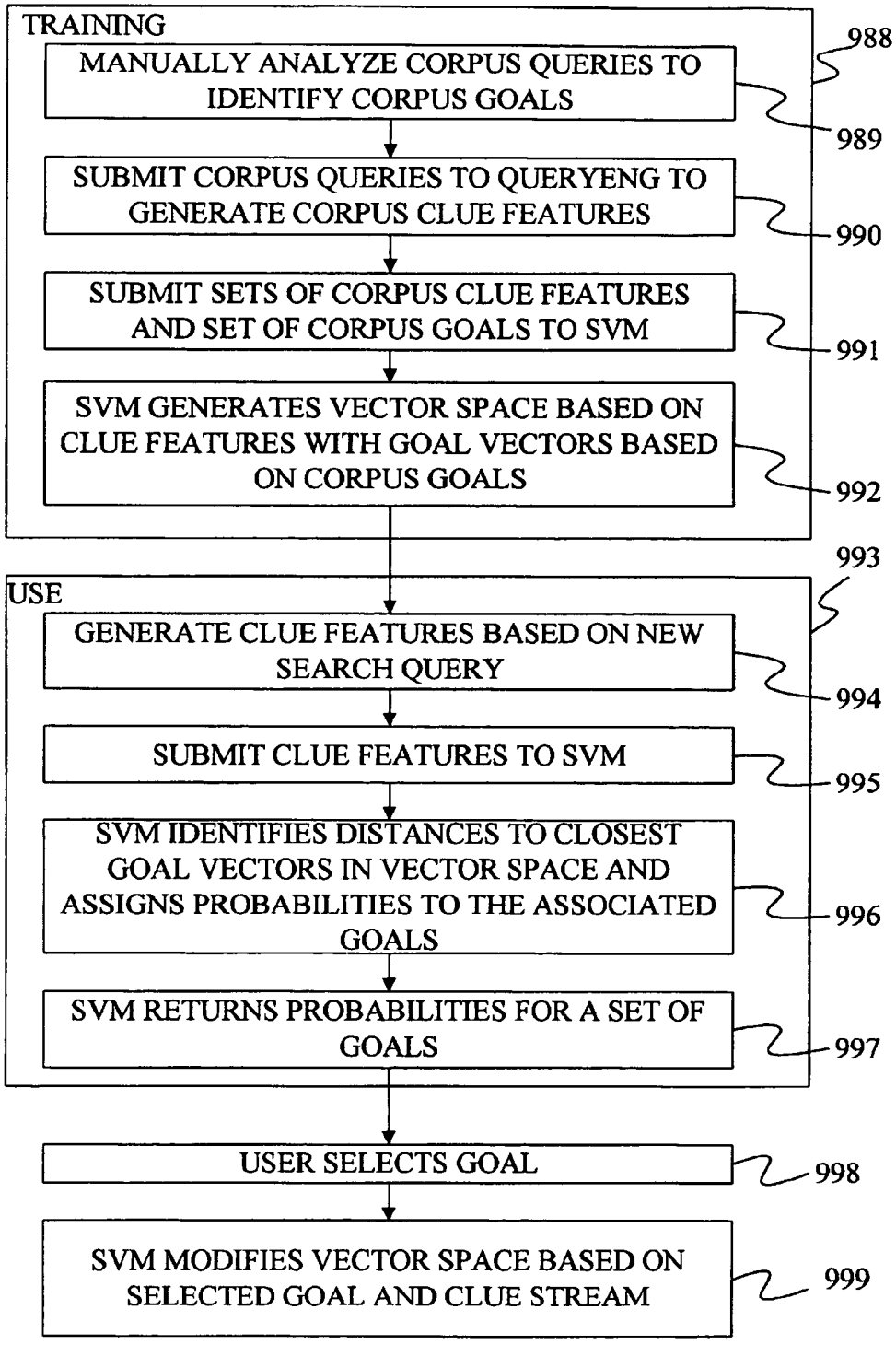

TRAINING
MANUALLY ANALYZE CORPUS QUERIES TO IDENTIFY CORPUS GOALS
SUBMIT CORPUS QUERIES TO QUERYENG TO GENERATE CORPUS CLUE FEATURES
SUBMIT SETS OF CORPUS CLUE FEATURES AND SET OF CORPUS GOALS TO SVM
SVM GENERATES VECTOR SPACE BASED ON CLUE FEATURES WITH GOAL VECTORS BASED ON CORPUS GOALS
USE
GENERATE CLUE FEATURES BASED ON NEW SEARCH QUERY
SUBMIT CLUE FEATURES TO SVM
SVM IDENTIFIES DISTANCES TO CLOSEST GOAL VECTORS IN VECTOR SPACE AND ASSIGNS PROBABILITIES TO THE ASSOCIATED GOALS
SVM RETURNS PROBABILITIES FOR A SET OF GOALS
USER SELECTS GOAL
SVM MODIFIES VECTOR SPACE BASED ON SELECTED GOAL AND CLUE STREAM
988
989
990
991
992
993
994
995
996
997
998
999

FIG. 17

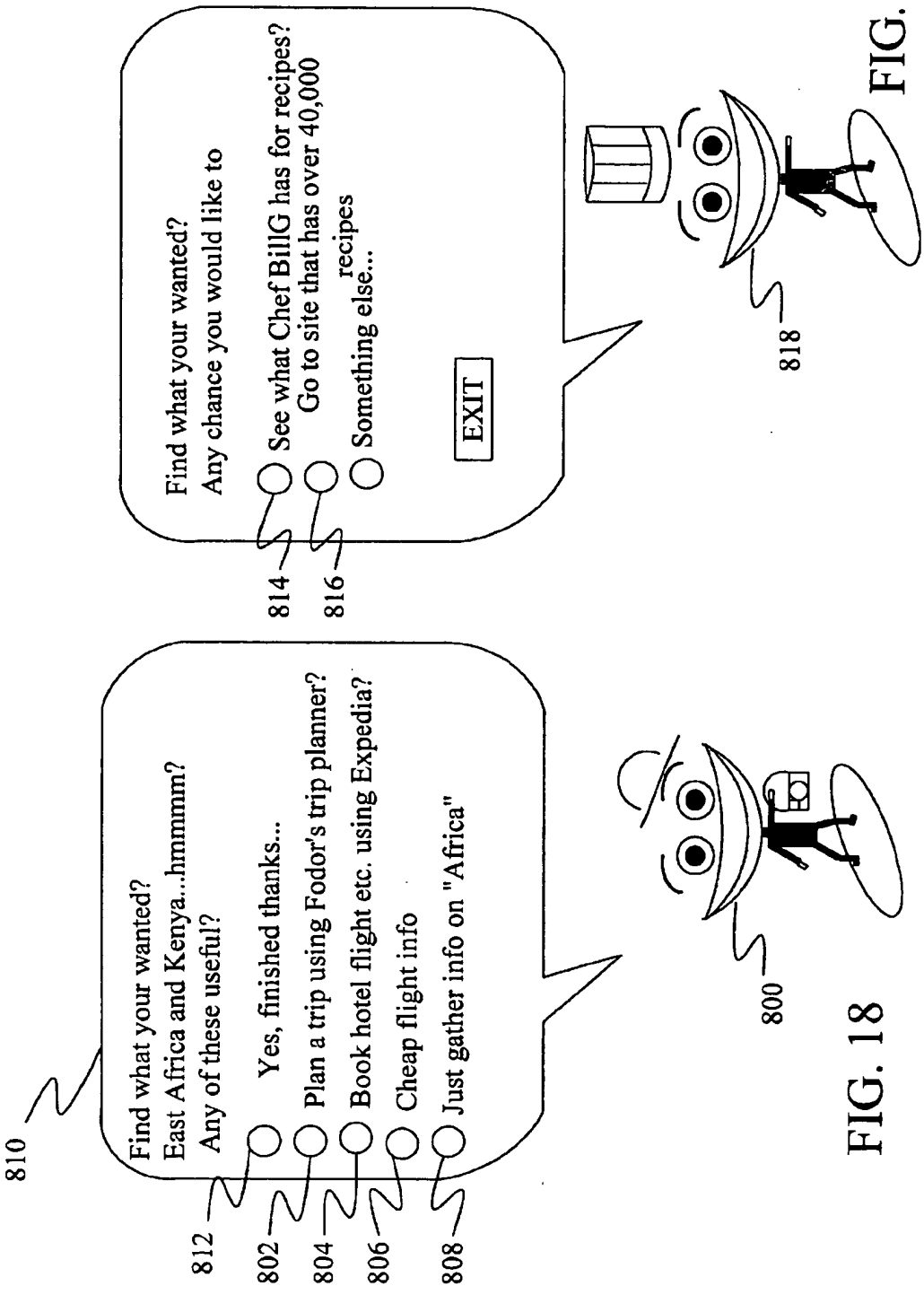

Find what your wanted?
East Africa and Kenya...hmmmm?
Any of these useful?
Yes, finished thanks...
Plan a trip using Fodor's trip planner?
Book hotel flight etc. using Expedia?
Cheap flight info
Just gather info on "Africa"
810
812
802
804
806
808
800
Find what your wanted?
Any chance you would like to
See what Chef BillG has for recipes?
Go to site that has over 40,000 recipes
Something else...
EXIT
814
816
818

FIG. 18

FIG. 19

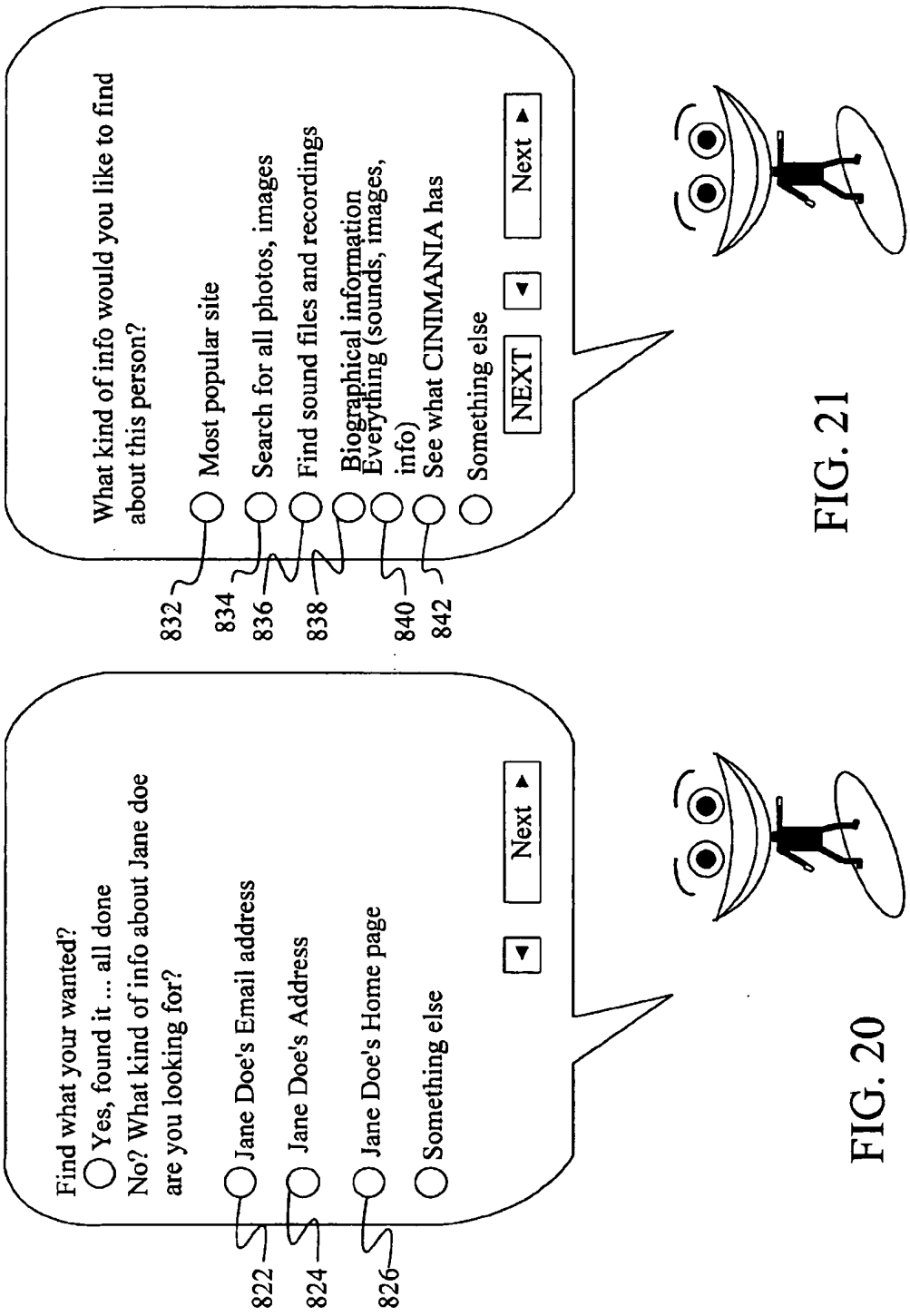

Find what your wanted?
Yes, found it ... all done
No? What kind of info about Jane doe are you looking for?
Jane Doe's Email address
Jane Doe's Address
Jane Doe's Home page
Something else
Next
822
824
826
What kind of info would you like to find about this person?
Most popular site
Search for all photos, images
Find sound files and recordings
Biographical information
Everything (sounds, images, info)
See what CINIMANIA has
Something else
NEXT
Next
832
834
836
838
840
842

FIG. 20

FIG. 21

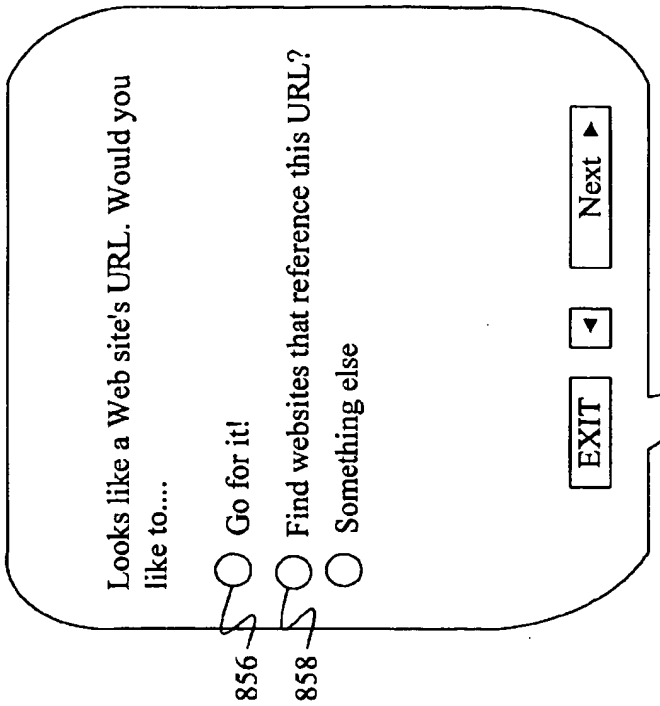

Looks like a Web site's URL. Would you like to....
Go for it!
Find websites that reference this URL?
Something else
EXIT
Next
856
858

FIG. 23

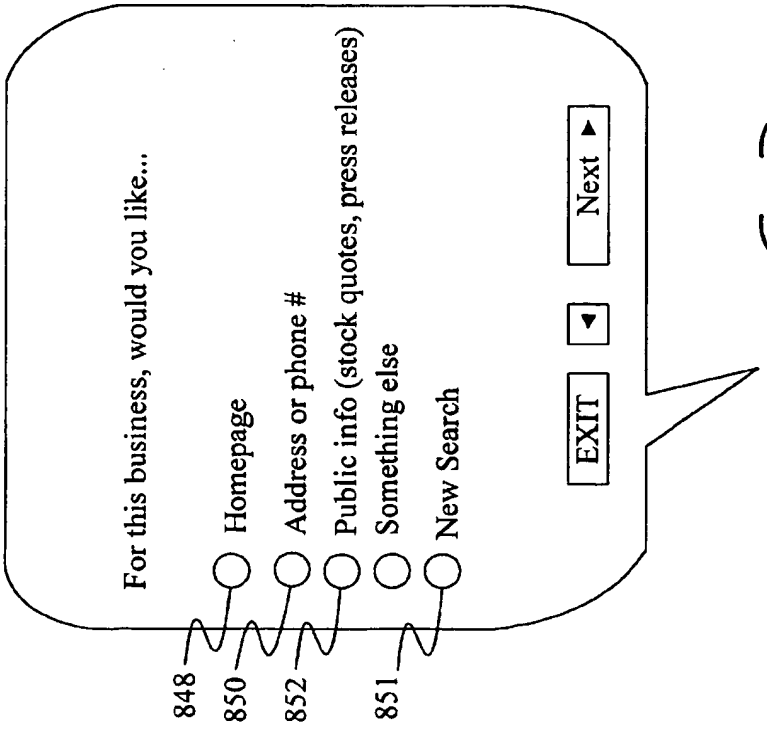

For this business, would you like...
Homepage
Address or phone #
Public info (stock quotes, press releases)
Something else
New Search
EXIT
Next
848
850
852
851

FIG. 22

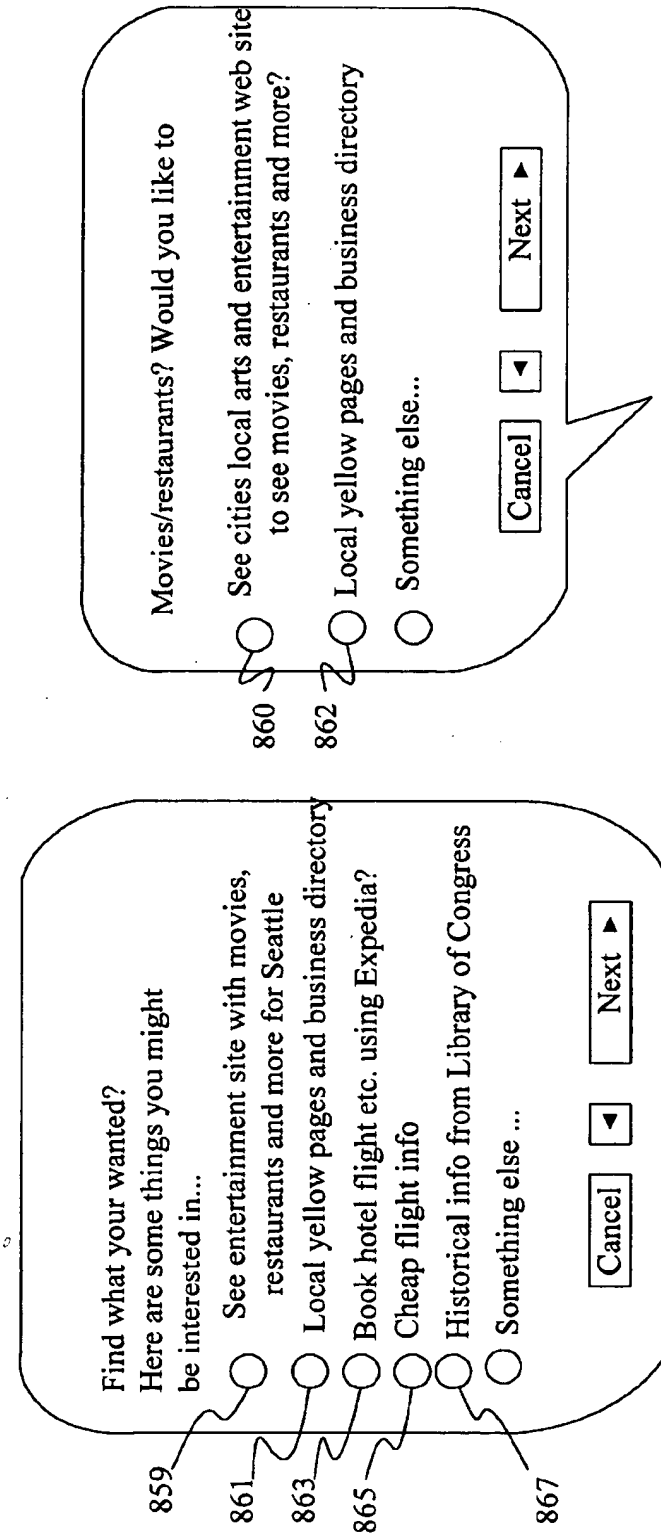

Movies/restaurants? Would you like to
See cities local arts and entertainment web site to see movies, restaurants and more?
Local yellow pages and business directory
Something else...
Cancel
Next
860
862

FIG. 25


Find what your wanted?
Here are some things you might be interested in...
See entertainment site with movies, restaurants and more for Seattle
Local yellow pages and business directory
Book hotel flight etc. using Expedia?
Cheap flight info
Historical info from Library of Congress
Something else ...
Cancel
Next
859
861
863
865
867

FIG. 24

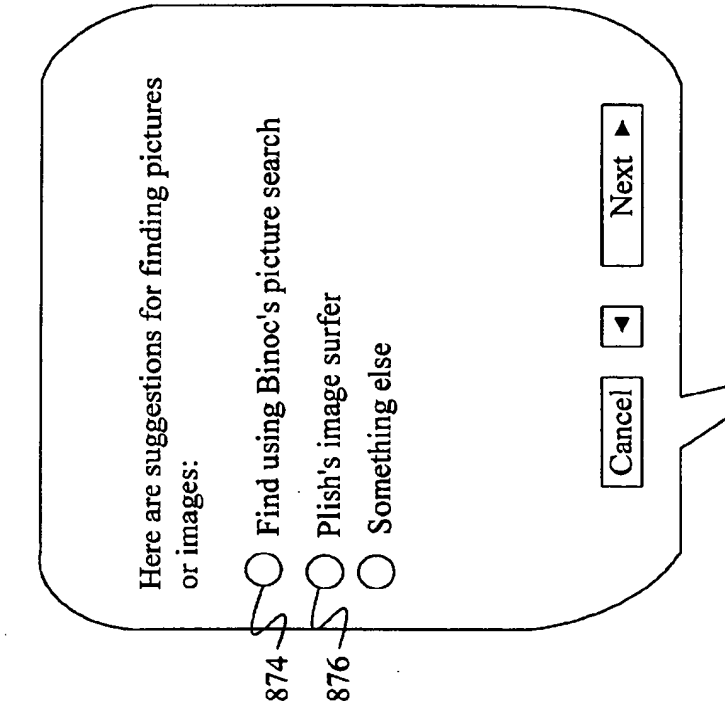

Here are suggestions for finding pictures or images:
Find using Binoc's picture search
Plish's image surfer
Something else
Cancel
Next
874
876

FIG. 27

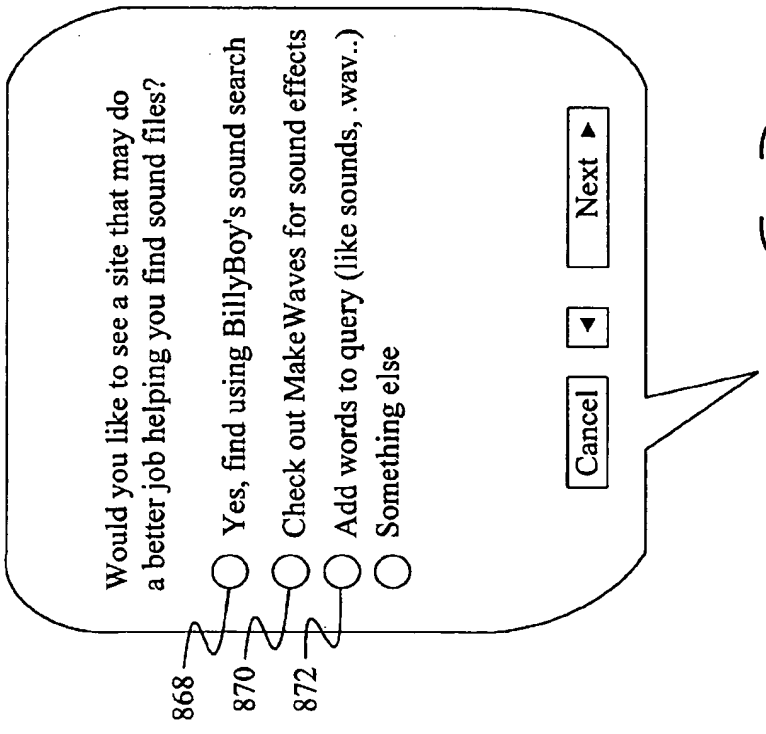

Would you like to see a site that may do a better job helping you find sound files?
Yes, find using BillyBoy's sound search
Check out MakeWaves for sound effects
Add words to query (like sounds, .wav..)
Something else
Cancel
Next
868
870
872

FIG. 26

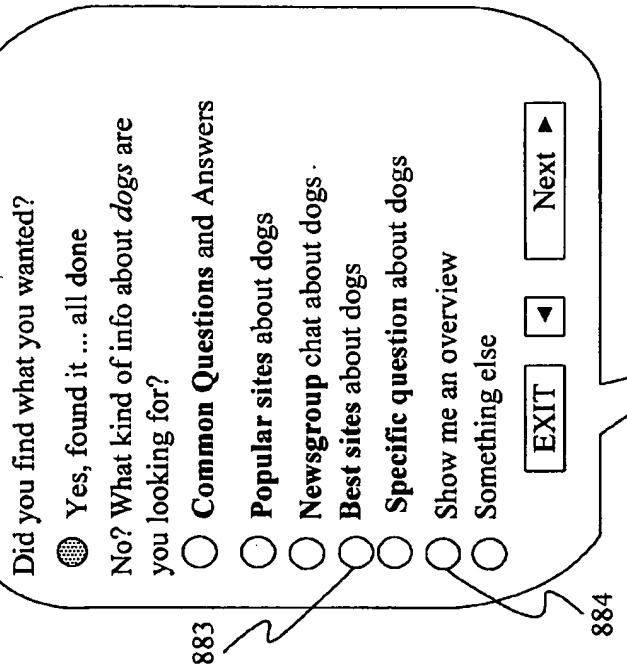

Did you find what you wanted?
Yes, found it ... all done
No? What kind of info about dogs are you looking for?
Common Questions and Answers
Popular sites about dogs
Newsgroup chat about dogs
Best sites about dogs
Specific question about dogs
Show me an overview
Something else
EXIT
Next
883
884

FIG. 29

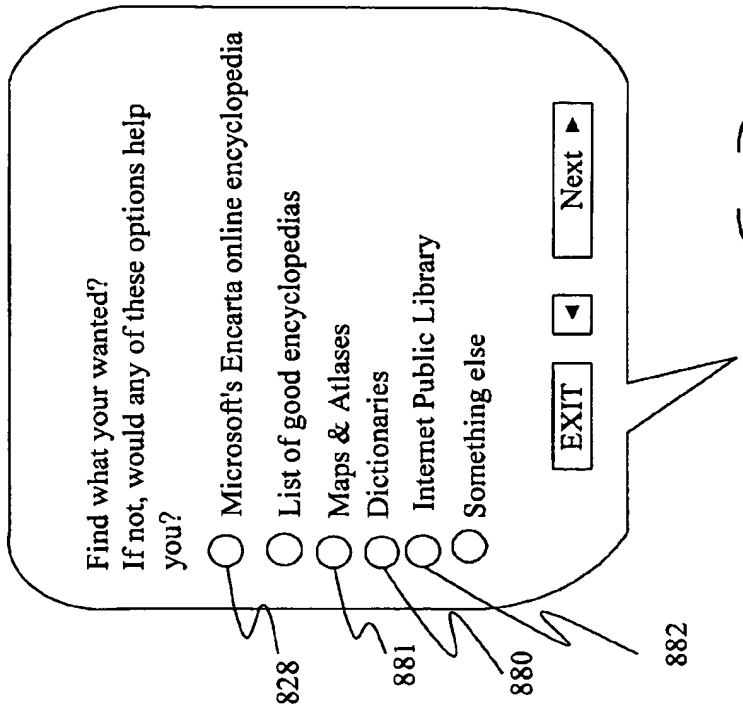

Find what your wanted?
If not, would any of these options help you?
Microsoft's Encarta online encyclopedia
List of good encyclopedias
Maps & Atlases
Dictionaries
Internet Public Library
Something else
EXIT
Next
828
881
880
882

FIG. 28

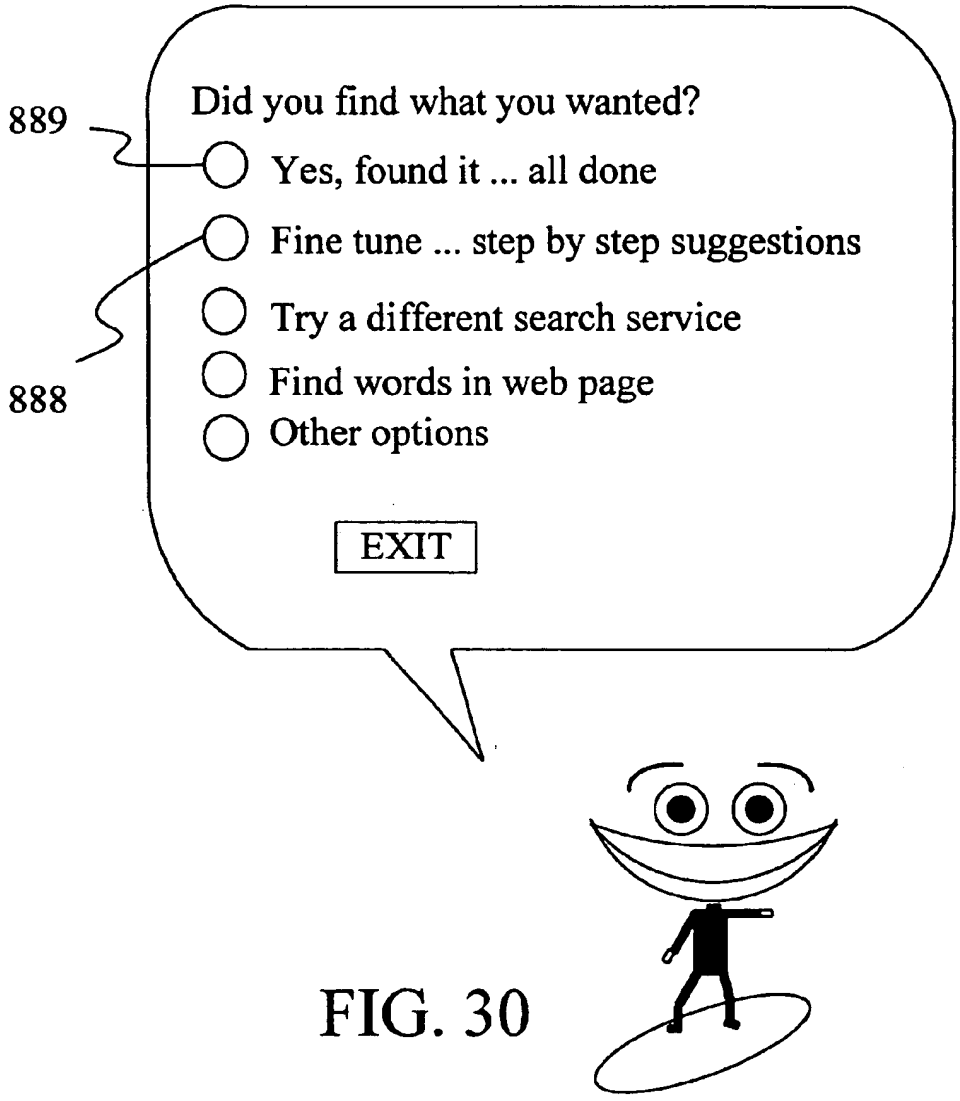

889
888
Did you find what you wanted?
Yes, found it ... all done
Fine tune ... step by step suggestions
Try a different search service
Find words in web page
Other options
EXIT

FIG. 30

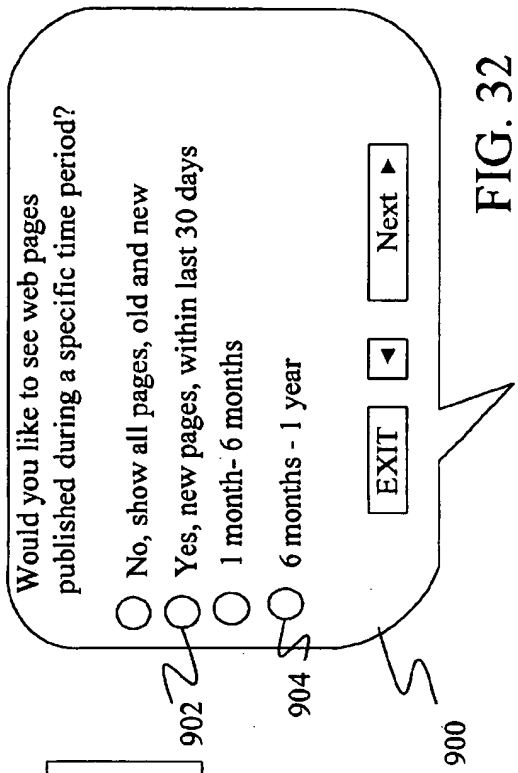



FIG. 32

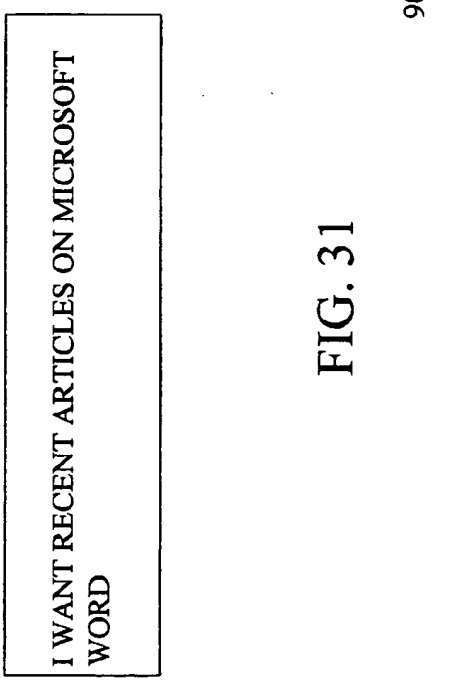



FIG. 31

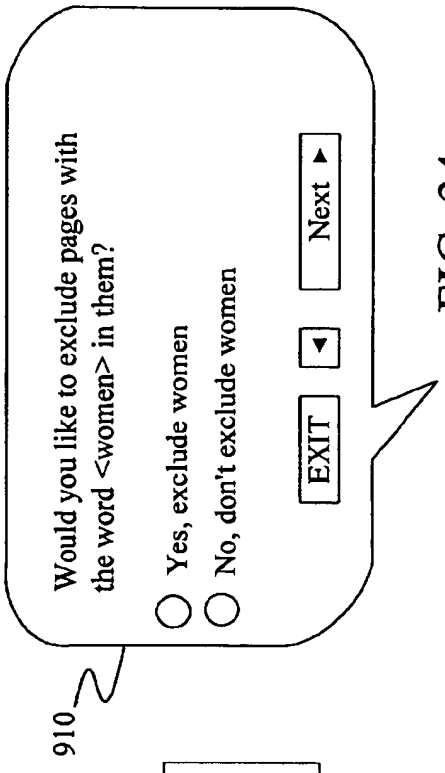



FIG. 34


FIG. 33

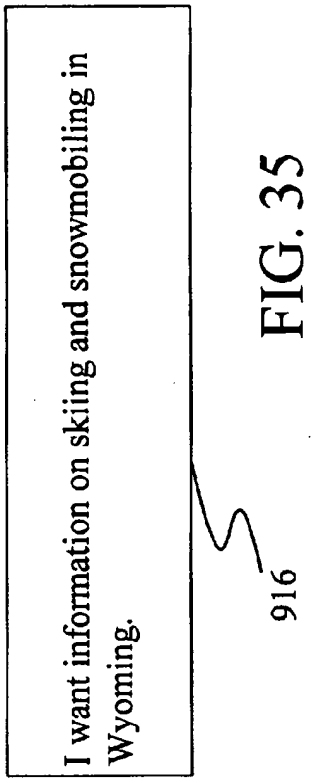

I want information on skiing and snowmobiling in Wyoming.
916

FIG. 35

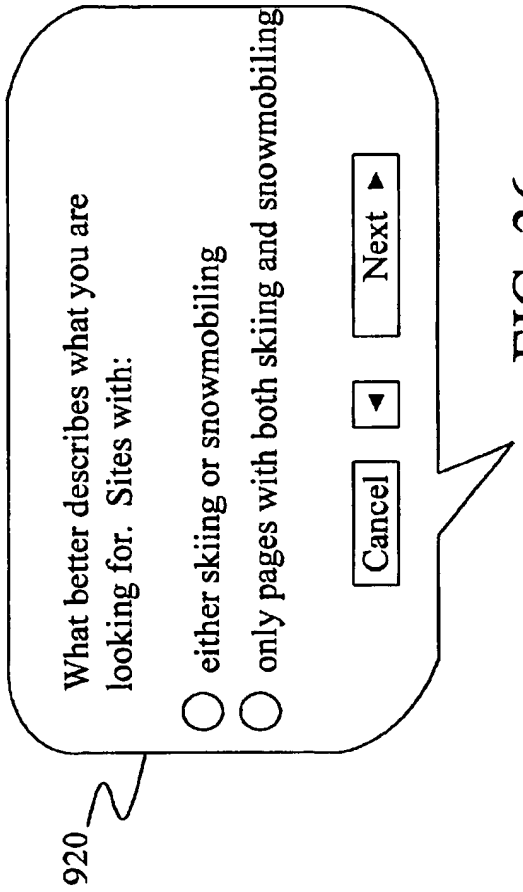

What better describes what you are looking for. Sites with:
either skiing or snowmobiling
only pages with both skiing and snowmobiling
Cancel
Next
920

FIG. 36

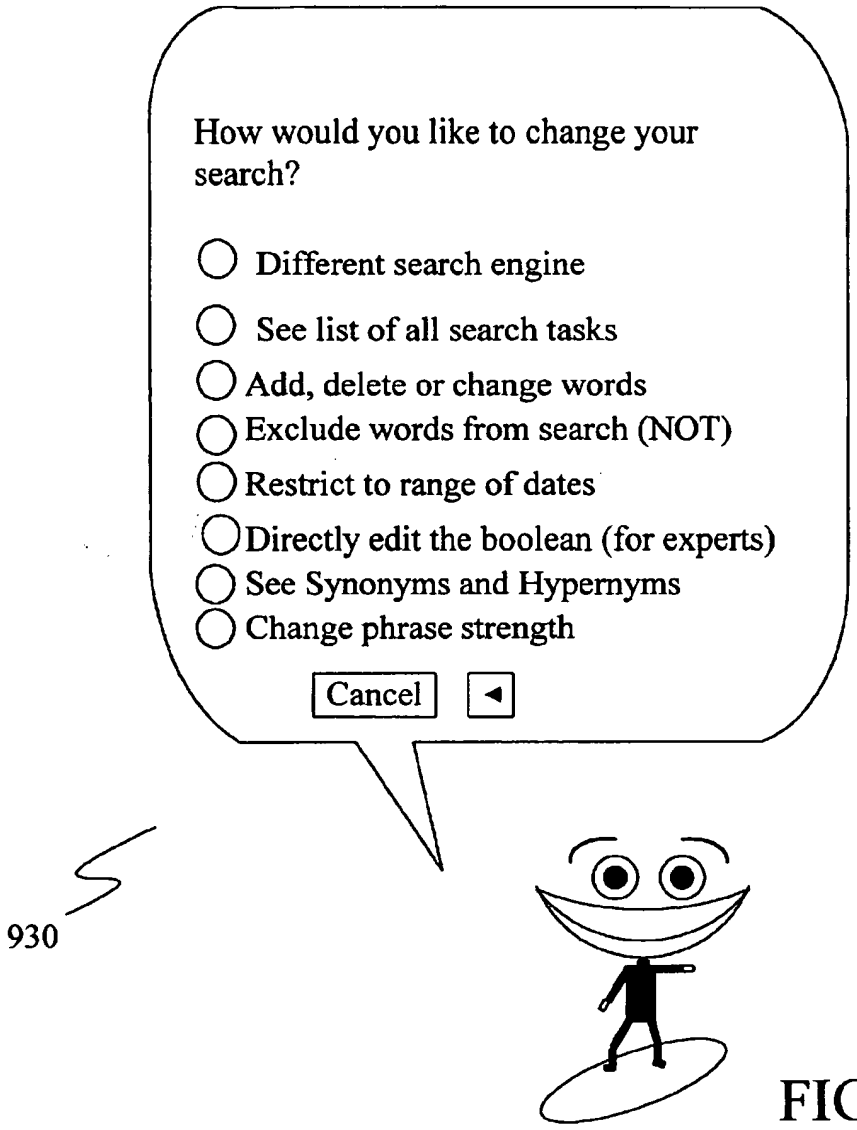

How would you like to change your search?
Different search engine
See list of all search tasks
Add, delete or change words
Exclude words from search (NOT)
Restrict to range of dates
Directly edit the boolean (for experts)
See Synonyms and Hypernyms
Change phrase strength
Cancel
930

FIG. 37

# SYSTEM FOR IMPROVING SEARCH AREA SELECTION

## RELATED APPLICATIONS

The present application is related to three applications filed on the same date herewith that are respectively entitled and have serial numbers of SYSTEM FOR ENHANCING A QUERY INTERFACE, Ser. No. 09/221,663; SYSTEM FOR IMPROVING SEARCH TEXT, Ser. No. 09/221,659; and COMPUTERIZED SEARCHING TOOL WITH SPELL CHECKING, Ser. No. 09/221,028.

## BACKGROUND OF THE INVENTION

The present invention relates to searching a network for information. In particular, the present invention relates to search tools used in computer searching.

Computer networks connect large numbers of computers together so they may share data and applications. Examples include Intranets that connect computers within a business or institution and the Internet, which connects computers throughout the world.

A single computer can be connected to both an Intranet and the Internet. In such a configuration, the computer can use data and applications found on any of its own storage media such as its hard disc drive, its optical drive, or its tape drive. It can also use data and applications located on another computer connected to the Intranet or Internet. Given the large number of locations from which a computer can extract data and the increasing amount of storage capacity at each of these locations, users have found it increasingly difficult to isolate the information they desire.

In recent years, users have begun to use search engines to help them search the Internet. Typically, search engines accept a search query from the user and then look for the search query's terms in an indexed list of terms. The indexed list is generated by parsing text found on individual Internet pages and indexing the text by the page's Uniform Resource Locator (URL).

Since it is impossible to index every page on the Internet, each search engine selects a set of pages to index. Since each search engine is created by a different group of people, different search engines index different sets of pages. In fact, some search engines have become extremely specialized and only index pages related to a specific category of information such as sports or celebrities.

In addition, different search engines search through their index in different ways and are optimized using different query structures. Some search engines are optimized to accept free-text queries. Others are optimized to accept queries with logical operators such as "AND" and "OR".

The differences between various search engines are largely unknown by average computer users. Therefore, they are not able to determine which search engine would best suit their searching goals. In addition, many of the specialized search engines that index specific categories of pages are unknown to average computer users. Therefore, users are not fully utilizing the variety of search engines available on the Internet.

Currently, there are no tools available to help computer users identify which search engines they should be using to optimize their search.

## SUMMARY OF THE INVENTION

A method of aiding a user in searching a computer environment includes retrieving a search query from a user, accessing a user profile and selecting a search area based on the search query and the user profile.

In other embodiments of the present invention, a method provides for receiving a search query, categorizing at least one term in the search query based on an indexed list of terms derived from pages on a network, and identifying at least one search area based at least in part on the category of a search term. In still further embodiments, the search area is selected based on a combination of the user profile and the category of the search term.

Other aspects of the present invention include identifying the scope of a search based on a search query retrieved from a user and using the scope to identify a search area. The scope relates to the level of detail that the user wants the returned documents to provide. The scope is identified in a number of different ways. In one embodiment, the scope is identified based on the number of words in the search query. In other embodiments, the scope is determined by comparing the search query against a user profile. In still other embodiments, the number of terms and the user profile are combined to identify the search scope.

An embodiment of the present invention also provides a method that categorizes search tools based on their ability to search using search queries of selected formats. The method then sends a search query to a search tool based on the search query's format and the category of the search tool.

## BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a plan view of a computing environment of the present invention.

FIG. **2** is a block diagram of an architecture of an embodiment of the present invention.

FIG. **3** is a flow diagram describing the initial processes of an embodiment of the present invention.

FIG. **4A** is an example of an initial display produced by an embodiment of the present invention.

FIG. **4B** is an example of an additional display produced by an embodiment of the present invention.

FIG. **5** is an example display produced by the present invention if a user wishes to go to a previous site.

FIG. **6** is an example text display with an animated character in accordance with an aspect of the present invention shown in conjunction with an Internet browser window.

FIG. **7A** is an example display produced by the present invention when a user wants to enter a new search.

FIG. **7B** is an alternative example display produced by the present invention when a user wants to enter a new search.

FIG. **7C** is an example display produced by the present invention showing spell-checking options provided by an embodiment of the present invention.

FIG. **8** is a flow diagram of the central process of an embodiment of the present invention.

FIG. **9** is a flow diagram showing a process for performing a natural language parse under an embodiment of the present invention.

FIG. **10** is a flow diagram for making a remote call to an object located on a remote server under an embodiment of the present invention.

FIG. **11** is a layout for an NLP block produced by a NLP component under an embodiment of the present invention.

FIG. **12** is an example of a layout for the NLP data of one search term in the NLP block.

FIG. **13** is a flow diagram of a process for identifying possible topics under an embodiment of the present invention.

FIGS. 14A and 14B are flow diagrams of a process followed by a Topic Dictionary component under an embodiment of the present invention.

FIG. 14C is a block diagram of components used in connection with the Topic Dictionary component.

FIG. 15 is a flow diagram for constructing a Boolean search query based on NLP data under an embodiment of the present invention.

FIG. 16 is a flow diagram for submitting a search query to a search area under an embodiment of the present invention.

FIG. 17 is a flow diagram for training and using the support vector machine of FIG. 2.

FIG. 18 is an example web companion display produced in response to a search query directed toward a country or continent.

FIG. 19 is an example web companion display produced in response to a search query directed toward food.

FIG. 20 is an example web companion display produced in response to a search query directed toward a non-famous person's name.

FIG. 21 is an example web companion display produced in response to a search query directed toward a famous person's name.

FIG. 22 is an example web companion display produced in response to a search query directed toward a company name.

FIG. 23 is an example web companion display produced in response to a search query directed toward an URL.

FIG. 24 is an example web companion display produced in response to a search query directed toward a city.

FIG. 25 is an example web companion display produced in response to a search query directed toward a restaurant.

FIG. 26 is an example web companion display produced in response to a search query directed toward sound.

FIG. 27 is an example web companion display produced in response to a search query directed toward pictures.

FIG. 28 is an example web companion display produced in response to a search query having a narrow scope.

FIG. 29 is an example web companion display produced in response to a search query having a broad scope.

FIG. 30 is an example web companion display produced to provide alternative search suggestions.

FIG. 31 is an example of a search query with an ambiguity as to time.

FIG. 32 is an example of a web companion display produced to remove an ambiguity related to time.

FIG. 33 is an example of a search query with an exclusion ambiguity.

FIG. 34 is an example of a web companion display produced to remove an exclusion ambiguity.

FIG. 35 is an example of a search query with a coordinating structure ambiguity.

FIG. 36 is an example of a web companion display produced to remove a coordination structure ambiguity.

FIG. 37 is an example of a web companion display produced to fine tune the search query if it does not contain ambiguities.

## DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

FIG. 1 and the related discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. Although not required, the invention will be described, at least in part, in the general context of computer-executable instructions, such as program modules, being executed by a personal computer. Generally, program modules include routine programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

With reference to FIG. 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a conventional personal computer 20, including a processing unit (CPU) 21, a system memory 22, and a system bus 23 that couples various system components including the system memory 22 to the processing unit 21. The system bus 23 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory 22 includes read only memory (ROM) 24 and random access memory (RAM) 25. A basic input/output (BIOS) 26, containing the basic routine that helps to transfer information between elements within the personal computer 20, such as during start-up, is stored in ROM 24. The personal computer 20 further includes a hard disk drive 27 for reading from and writing to a hard disk (not shown), a magnetic disk drive 28 for reading from or writing to removable magnetic disk 29, and an optical disk drive 30 for reading from or writing to a removable optical disk 31 such as a CD ROM or other optical media. The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, magnetic disk drive interface 33, and an optical drive interface 34, respectively. The drives and the associated computer-readable media provide nonvolatile storage of computer readable instructions, data structures, program modules and other data for the personal computer 20.

Although the exemplary environment described herein employs the hard disk, the removable magnetic disk 29 and the removable optical disk 31, it should be appreciated by those skilled in the art that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMS), read only memory (ROM), and the like, may also be used in the exemplary operating environment.

A number of program modules may be stored on the hard disk, magnetic disk 29, optical disk 31, ROM 24 or RAM 25, including an operating system 35, one or more application programs 36, other program modules 37, and program data 38. A user may enter commands and information into the personal computer 20 through local input devices such as a keyboard 40, pointing device 42 and a microphone 43. Other input devices (not shown) may include a joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 21 through a serial port interface 46 that is coupled to the system bus 23, but may be connected by other interfaces, such as a sound card, a parallel port, a game port or a

universal serial bus (USB). A monitor **47** or other type of display device is also connected to the system bus **23** via an interface, such as a video adapter **48**. In addition to the monitor **47**, personal computers may typically include other peripheral output devices, such as a speaker **45** and printers (not shown).

The personal computer **20** may operate in a networked environment using logic connections to one or more remote computers, such as a remote computer **49**. The remote computer **49** may be another personal computer, a hand-held device, a server, a router, a network PC, a peer device or other network node, and typically includes many or all of the elements described above relative to the personal computer **20**, although only a memory storage device **50** has been illustrated in FIG. **1**. The logic connections depicted in FIG. **1** include a local area network (LAN) **51** and a wide area network (WAN) **52**. Such networking environments are commonplace in offices, enterprise-wide computer network Intranets, and the Internet.

When used in a LAN networking environment, the personal computer **20** is connected to the local area network **51** through a network interface or adapter **53**. When used in a WAN networking environment, the personal computer **20** typically includes a modem **54** or other means for establishing communications over the wide area network **52**, such as the Internet. The modem **54**, which may be internal or external, is connected to the system bus **23** via the serial port interface **46**. In a network environment, program modules depicted relative to the personal computer **20**, or portions thereof, may be stored in the remote memory storage devices. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used. For example, a wireless communication link may be established between one or more portions of the network.

The present invention provides a web companion that acts as an interactive searching aid for searching a computer environment, especially an environment that includes an Intranet or the Internet. The web companion is interactive in the sense that it provides the user with searching options based on the search query provided by the user and previous searching options the user has selected. Some of the options provided by the web companion are possible search goals that the user may have, such as a person's e-mail address, or photographs of a celebrity. If the user selects one of the goals, the web companion can automatically select an appropriate search area and/or adjust the user's search query to improve the likelihood that the user will find what they are looking for.

The web companion may be invoked in a number of different ways. In a Windows 95®, Windows 98® or Windows NT® based operating system provided by Microsoft Corporation, the web companion can be invoked by "double-clicking" on an icon appearing in the environment. In addition, the web companion can be invoked from within a browser such as Internet Explorer 4 (IE4) from Microsoft Corporation. In particular, the web companion can be registered with IE4 so that IE4 opens the web companion in the background when IE4 is opened. In such a configuration, the web companion does not display an interface while it is operating in the background. When the user enters a search in IE4, either through a search engine on the Internet or through the browser's search screen, the search is provided to the web companion. The web companion then processes the search through steps described below and determines possible suggestions that would aid the user. In some embodiments, the web companion then generates an interface to display the suggestions to the user as described below. In other embodiments, the web companion only displays an interface if the suggestions have a high probability of being helpful to the user. When the web companion is invoked through IE4 in this manner, the web companion display disappears if the user does not adopt a suggestion made by the web companion. The web companion may also be stored on a remote server and invoked through a network connection to the remote server.

FIG. **2** shows a component architecture for the present invention. The web companion is initiated by calling an executable application identified as WEB COMPANION **200** in FIG. **2**. WEB COMPANION **200** invokes an instance of IE4 control **202**, which is an extendable hypertext markup language (html) interpreter produced by Microsoft Corporation. WEB COMPANION **200** also passes a .htm page denoted as DEFAULT.HTM **204** to IE4 control **202**, thereby causing IE4 control **202** to execute the instructions in DEFAULT.HTM **204**.

The instructions in DEFAULT.HTM **204** include requests for instances of three ACTIVE-X controls: SEARCH-AGENT **206**, QUERYENG **208**, and TRUEVOICE **210**. Each ACTIVE-X control includes methods that can be invoked by DEFAULT.HTM **204** and each ACTIVE-X control is able to fire events that are trapped by DEFAULT.HTM **204**.

QUERYENG **208** cooperates with DEFAULT.HTM **204** and WEB COMPANION **200** to perform most of the functions of the present invention. SEARCH-AGENT **206** generates, positions and animates a graphical character, shown as character **262** in FIG. **4B**, based on method calls from DEFAULT.HTM **204**. SEARCH-AGENT **206** also allows the user to move the animated character using an input device. When the animated character is moved by the user, SEARCH-AGENT **206** fires an event indicating the new position of the character, which is trapped by DEFAULT.HTM **204**.

TRUEVOICE **210** produces sounds based on method calls made by DEFAULT.HTM **204**. Typically, these sounds are timed to coincide with the animation of the character produced by SEARCH-AGENT **206**.

WEB COMPANION **200** generates a balloon, such as balloon **260** of FIG. **4B**. The balloon is positioned on the screen based on the location of the animated character, which is provided to WEB COMPANION **200** by QUERYENG **208**. Based on instructions in DEFAULT.HTM **204** or alternatively, instructions in Active Server Pages (.ASP) called by DEFAULT.HTM **204**, IE4 control **202** displays text and control buttons in the balloon. An example of text displayed by IE4 control **202** is shown in FIG. **4B** as text **261** along with an example of a control button **263**. Control button **263** may be activated by the user by positioning the cursor over the button and pressing an input device button.

The Active Server Pages called by DEFAULT.HTM include HTML instructions. Although only three ASP files **212**, **214** and **216** are shown in FIG. **2**, those skilled in the art will recognize that any number of .ASP files may be used in conjunction with DEFAULT.HTM **204**.

FIG. **3** is a flow diagram of the steps followed by the computer-executable instructions found in WEB COMPANION **200**, IE4 control **202**, DEFAULT.HTM **204**, SEARCH-AGENT **206**, and QUERYENG **208**. In an initial step **229**, DEFAULT.HTM determines if this is the first time WEB COMPANION **200** has been invoked by this user. If it is the first invocation by this user, an introductory interface is provided at step **231** as shown in FIG. **4A**. In FIG. **4A**, IE4

control 202 displays introductory text 265, produced by DEFAULT.HTM 204, in a balloon 267 produced by WEB COMAPANION 200. At the same time, SEARCH-AGENT 206 displays an animated character 269 next to the introductory balloon.

If this is not the first invocation of WEB COMPANION 200, or after the display of the initial screen, the process continues at step 228 where a first selection display is produced by WEB COMPANION 200, DEFAULT.HTM 204 AND SEARCH-AGENT 206. An example of this display is shown in FIG. 4B with a character 262 produced by SEARCH-AGENT 206 shown next to a balloon 260 produced by WEB COMPANION 200 that contains text 261 and control buttons 263 produced by DEFAULT.HTM 204 and IE4 control 202. In the selection display of FIG. 4B, the user may either choose to perform a new search or go to a previously visited site. Thus, depending on what the user selects, the process either continues at step 230 OR 246.

If the user chooses to go to a previous site, the computer-executable instructions follow step 230 to step 232, where they locate recently visited sites stored for this user. In one embodiment, the recently visited sites are stored in Registry 222 of FIG. 2, which is a memory location maintained by many of the operating systems produced by Microsoft Corporation. However, the recently visited sites may be stored in any suitable memory location on the local machine or a server. After locating the names of recently visited sites, the computer-executable instructions proceed to step 234, where the instructions locate the names of sites that the user frequently visits. In one embodiment, these sites are also stored in Registry 222.

At step 236, DEFAULT.HTM 204 causes IE4 control 202 to display a selectable list of recently visited sites and frequently visited sites. An example of such a selectable list is shown in FIG. 5 in balloon 264. The selectable list is accompanied by animated character 266, which is produced by SEARCH-AGENT 206.

The selectable list of balloon 264 includes selectable entries for five recently visited sites 268, 270, 272, 274, and 276, and selectable entries for five frequently visited sites 278, 280, 282, 284, and 286. The selectable list also includes an option to search the Internet. In many embodiments, the names of the sites that appear in balloon 264 are the common names for the sites. In other words, the Uniform Resource Locators (URLs) for the sites normally do not appear in balloon 264, since most users find it difficult to associate a site's URL with its contents. However, to accommodate users that want to see a site's URL, the present invention provides a pop-up window that appears if the user pauses the display caret over a site's name. An example of this is shown in FIG. 5, where URL window 280 has opened for entry 270. In FIG. 5, the caret is not shown so that entry 270 is not obscured.

While the selectable list of balloon 264 is displayed, DEFAULT.HTM 204 waits for the user to select one of the listed sites in a step 237. If the user selects a site, the computer-executable instructions follow step 238 to step 240.

In step 240, DEFAULT.HTM 204 calls a method in QUERYENG 208 to pass a message to WEB COMPANION 200, asking WEB COMPANION 200 to locate or instantiate an Internet browser such as IEXPLORE from Microsoft Corporation. If one or more Internet browsers are open, WEB COMPANION 200 selects the top browser. If there are no open browsers, WEB COMPANION 200 opens a browser. In FIG.2, the opened browser is shown as IEXPLORE 218. DEFAULT.HTM 204 passes the URL of the selected site through QUERYENG 208 and WEB COMPANION 200 to IEXPLORE 218 at step 242.

IEXPLORE 218 uses the site's URL to locate the site's server over a network connection, such as the Internet, and to make a request from the server for the site's content. The located server, shown as server 219 in FIG. 2, returns the requested content to IEXPLORE 218. As those skilled in the art will recognize, the returned content can take many forms. IEXPLORE 218 determines the form of the content it receives from server 219 and displays the content in a browser window. IEXPLORE 218 remains open until the user closes the browser window. This allows the user to perform further Internet searching and viewing operations through the browser. Such operations are separate and independent of the operation of the web companion.

FIG. 6 presents a screen display where a web companion balloon 300 and a character 304 appear on the same screen as an Internet browser window 306 created through the steps described above. Browser window 306 is independent of balloon 300 and character 304 and may be moved, expanded, closed, and have its dimensions changed independently of balloon 300 and character 304.

If at steps 228 or 237 of FIG. 3, the user selects to perform a new search, the computer-executable instructions continue at step 246. Step 246 leads to step 320 of an additional flow diagram shown in FIG. 8.

At step 320 of FIG. 8, DEFAULT.HTM 204 causes IE4 control 202 to display a search interface. An example of such a search interface is shown in FIG. 7A, where the interface appears within a balloon 308 produced by WEB COMPANION 200 that appears adjacent animated character 310 produced by SEARCH-AGENT 206.

In addition to defining the search interface shown in FIG. 7A, DEFAULT.HTM 204 establishes an instance of a spell checking object identified as SPELLCHECK 221 in FIG. 2. DEFAULT.HTM 204 assigns a text box 312 in balloon 308 to SPELLCHECK 221 so that text entries and cursor movements within text box 312 are passed directly to SPELLCHECK 221. This allows SPELLCHECK 221 to verify the spelling of words as they are entered by the user and to suggest alternative spellings when the user places the cursor over a word and activates a button on their mouse or track-ball.

The search interface found in balloon 308 of FIG. 7A includes a solicitation to the user to type in their search request in a natural language or free text format. In these formats, the user simply enters normal statements or questions and does not need to include logical operators to indicate the relationship between the terms of the search query. Text box 312 displays the user's search query as the user types and allows the user to modify their query. This search solicitation process is represented by step 320 of FIG. 8.

FIG. 7B provides an alternative search solicitation display to that shown in FIG. 7A. In FIG. 7B, a pull-down text box 250 is provided to accept and display the user's search text. Pull-down text box 250, includes a pull-down activation arrow 251 that causes a pull-down window 252 to be displayed when activated. Pull-down window 252 displays a selectable list of past search queries entered by the user and allows the user to select a past search query by highlighting it. Typically, past search queries are stored in Registry 222 of FIG. 2. However, they may be stored in any suitable memory location.

By recording the user's past searches and by allowing the user to review their past searches, the present invention

improves searching efficiency by reducing the likelihood that the user will unknowingly reuse unsuccessful searches or waste time trying to remember past successful searches.

While the user is entering their search query, the query is spell checked by SPELLCHECK **221** at a step **322** of FIG. **8**. If the search query includes a misspelled word, SPELLCHECK **221** provides a visual cue to the user that a word is misspelled. In many embodiments, this visual cue is a red line underneath the misspelled word. FIG. **7**A shows an example of a visual cue **309** beneath the misspelled word "amercan". In further embodiments, SPELLCHECK **221** displays a list of properly spelled words when the user activates a button on their input device. An example of such a display is shown in FIG. **7**C where a selectable list **311** is displayed by SPELLCHECK **221** in response to a button being activated on an input device while the cursor is positioned over the word "amercan". If the user selects one of the properly spelled words, SPELLCHECK **221** automatically replaces the misspelled word with the selected word.

Once the user has finished entering and modifying their query, they activate NEXT button **313** of FIG. **7**A or NEXT button **253** of FIG. **7**B, which causes the instructions of DEFAULT.HTM **204** to request the query text from SPELLCHECK **221** and to initiate processing of the query text. Such processing begins at step **324** of FIG. **8**, where the web companion performs a natural language parse (NLP) of the query text. The steps taken to perform the natural language parse are shown in detail in the flow diagram of FIG. **9**.

The NLP process of FIG. **9** begins at step **450**, where QUERYENG **208** of FIG. **2** replaces the spaces between words found in quotes in the user's query with underscores. At step **454**, the search query is stored in a shared buffer **223** of FIG. **2**. QUERYENG **208** then makes a call to invoke the NLP component at a step **456**.

The steps required to make the call to invoke the NLP component are shown in the flow diagram of FIG. **10**. The steps of FIG. **10** begin at step **480** where, as shown in FIG. **2**, WEB COMPANION **200** starts an instance of IEXPLORE **224**. WEB COMPANION **200** also passes a control file **225** to IEXPLORE **224**. In step **482**, control file **225** causes IEXPLORE **224** to start a second instance of QUERYENG denoted as QUERYENG **226** in FIG. **2**. QUERYENG **226** retrieves the search query stored in shared buffer **223** and packages the query to send it to the NLP component.

In step **486** of FIG. **10**, IEXPLORE **224** routes the package created by QUERYENG **226** to the NLP component. If the NLP component is on client **199**, the package is routed directly to the component. If the NLP component is located on a remote server, the package is routed to an Internet Server Application Programming Interface (ISAPI.DLL). The ISAPI.DLL then routes the package to the NLP component. In the embodiment of FIG. **2**, NLP component **227** is located on a remote server **233**, so the package is routed to an ISAPI.DLL **235**, which routes it to NLP component **227**. For clarity in the discussion below, NLP component **227** is used to describe the functions of the NLP component. However, it should be recognized that these functions are not dependent on the location of the NLP component and an NLP component with the same capabilities may alternatively be located on the client under the present invention.

In step **488**, the NLP component **227** performs natural language parsing functions on the search query. NLP component **227** uses known logical and syntactic rules to identify respective parts of speech for each term in the search query. NLP component **227** also identifies words that modify other terms in the search query and how words modify each other. In addition, NLP component **227** reduces each term in the search query to its most basic form and creates inflected and plural forms from the most basic form. NLP component **227** is also able to identify the semantics of certain words and categorize them. For instance, NLP component **227** is capable of recognizing that the term "recent" is related to time. Other categories include city, state, country, continent, and proper name, etc

NLP component **227** can also group together multiple words that represent a single conceptual item. For instance, NLP is able to identify the constituent parts of a date as belonging to a single date construct. To identify these "multi-word entries", NLP component **227** utilizes "factoids" and "captoids". Factoids are rules that identify multi-word entries on the basis of known facts. For example, NLP component **227** identifies "New Jersey" as a single multi-word entry because of the fact that New Jersey is a state. Captoids are rules that identify multi-word entries on the basis of the capitalization of terms in the query. For instance, if "Jack's Seafood Restaurant" is found in a search query, NLP component **227** will identify it as a multi-word entry on the basis of its capitalization.

NLP component **227** returns a block of NLP data embedded in an HTML page that is routed back to IEXPLORE **224**. This is shown in FIG. **10** as step **488**. At step **490**, IEXPLORE **224** replaces control file **225** with the HTML page returned by NLP component **227**. This causes QUERYENG **226** to close. At step **492**, the returned HTML page causes another instance of QUERYENG (QE3) to start, which at step **494** places the returned NLP block in shared buffer **223**. IEXPLORE **224** and QE3 then close at step **496**. The final step in making the call to NLP component **227** is step **498** where original QUERYENG **208** retrieves the returned NLP information from shared buffer **223**.

After the call to the NLP component the process of FIG. **9** continues at step **460**, where the NLP block returned by the NLP component is parsed into its constituent parts. One embodiment of the NLP block structure is shown in FIG. **11**, where NLP block **508** includes a data set for each NLP term. For example, NLP data for a first term is found in data set **510**, which is followed by a new-line marker (/N) **512**. The NLP data for the terms are together positioned between matching markers **514** and **516** that include lines of dashes ("-" ) that are terminated with new-line markers.

The NLP data for each term is normally of the form shown in FIG. **12** for data set **510**. Data set **510** includes nine fields: WORD POSITION **518**, WORD **520**, PART-OF-SPEECH **522**, WHAT-IT-MODIFIES **524**, HOW-IT-MODIFIES **526**, 'AND' or 'OR' SET **528**, PULRAL **530**, INFLECTED FORMS **532**, and NLP BITS **534**. WORD POSITION **518** contains the word's numerical location in the query and is in the form of an integer. WORD **520** and PART-OF-SPEECH **522** provide the word itself and its part-of-speech in the query, respectively. WHAT-IT-MODIFIES **524** indicates the number of any word that the current word modifies in the query and HOW-IT-MODIFIES **526** indicates the manner in which it modifies these other words. Examples of entries in HOW-IT-MODIFIES **526** include noun-adjective (NADJ) relationships where an adjective modifies a noun. It can also include generic modifying relationships such as the case where a noun modifies another noun, rather than an adjective modifying a noun. An example of this would be "Whitewater scandal" or "plant species". 'AND'-or-'OR' SET **528**

indicates whether the term is part of a coordinating set based on 'AND' or 'OR'. If the term is not part of such a set, the value in this field will be −1. If the term is part of an 'AND' set, the field will have a value between 0 and 99. If the term is part of an 'OR' set, this field will have a value greater than **100**.

PLURAL **530** provides a plural form of the term if appropriate and INFLECTED FORMS **532** provides any inflected forms of the term, separated from each other by commas. NLP BITS **534** provides semantic markers that indicate semantic information about the term. Examples of such markers include: "+tme" for terms related to time, "+city" for terms identifying a city, "+nme" for a person's name, "+neg" for a term providing a negative meaning, "+vulgar" for vulgar terms, and "+food" for terms related to food. The list above is only provided as an example and those skilled in the art will recognize that other markers are possible.

Returning to the flow diagram of FIG. **9**, the parsing function of step **460** parses the fields of each term into program variables used by QUERYENG **208** and DEFAULT.HTM **204**. When the parse is complete, any parsed words in the WORD field of the NLP block that are "stop words" are deleted to form a set of keywords. "Stop words" include words that occur so frequently in a language that they have no significance in a search query. Examples include articles such as "the" and "a", many prepositions, and common verbs such as "have" and "be". The removal of stop words is shown as step **462** in FIG. **9**. In one embodiment, stop words found in quoted phrases in the user's query and stop words that appear in a multi-word entry identified by NLP component **227** are not removed.

At step **464** of FIG. **9**, NLP data for each of the terms is checked to see if an inflected form returned by NLP matches the term itself or its plural form. If there is a match, the inflected form is deleted to remove redundancy in the NLP data. At step **466**, underscores are removed from between multi-word entries that appear as a single term in the returned NLP block. After step **466** of FIG. **9**, step **324** of FIG. **8** is complete and the conversion of the natural language parse data into keywords has been accomplished.

At step **326** of FIG. **8**, the keywords formed in step **324** are, if desired, used to modify the behaviors or animations of the character. For instance, in one embodiment of the invention, if one of the keywords is associated with the NLP vulgar bit, the character is changed so that it blushes.

In step **328** of FIG. **8**, the original search query, the keywords found in step **324** and their associated NLP bits are used to identify possible search topics. These search topics represent broad categories of information that the search query appears to be directed toward. The process of identifying these categories is shown in greater detail in the flow diagram of FIG. **13**.

In first step **549** of FIG. **13**, the keywords obtained in step **324** of FIG. **8** are stored in a shared buffer such as shared buffer **223** of FIG. **2**. In one embodiment, phrases that are in quotes in the user's query appear unchanged and remain in quotes in shared buffer **223**. A Topics Dictionary component is then called in step **550** using the technique described above for invoking the NLP component. To invoke the Topics Dictionary component using the technique described above, the control file **225** passed to IEXPLORE **224** is modified so that it causes the keywords to be passed to a Topics Dictionary component instead of the NLP component.

In the embodiment of FIG. **2**, a Topics Dictionary **239** is shown on server **233**. In other alternative embodiments, the Topics Dictionary is located on client **199** or on servers other than server **233**. Regardless of its location, Topics Dictionary **239** receives a package of keywords from IEXPLORE **224** and as shown in step **488** of FIG. **10**, performs functions on the terms in the package.

The operation of Topics Dictionary component **239** is shown through flow diagrams in FIGS. **14A** and **14B** and a block diagram in FIG. **14C**. The block diagram of FIG. **14C** shows the additional components utilized by Topics Dictionary component **239** to identify possible topics based on the keywords of the search text. The flow diagrams describe the process used by Topics Dictionary component **239** to identify the topics.

In an initial step **600** of FIG. **14A**, an executable denoted as WEB-PARSE **967** in FIG. **14B**, is initiated, which passes a URL list **960** to Topics Dictionary component **239**. URL list **960** includes a set of Uniform Resource Locators for pages located on the Internet and/or Intranet. In the list, each URL is associated with one or more topics and with a scripting function discussed further below. In step **601**, a database server **972**, which forms part of Topics Dictionary **239**, uses URL list **960** to generate a source database **961** that represents the associations found in URL list **960**.

At step **602**, WEB-PARSE **962** uses database server **972** to sequentially access the URL records stored in source database **961**. For each URL, WEB-PARSE **962** invokes a browser **964**, such as Internet Explorer **4** from Microsoft Corporation. Browser **964** uses the URL to retrieve the URL's page from a remote server **966** and to store the page locally.

Once the page has been retrieved, WEB-PARSE **962** calls scripting functions **963** that are associated with the URL in source database **961**. These scripting functions isolate desired information in the URL's page using dynamic object models of the HTML tags on the page. These object models are created by HTML object modeler **965** in response to method calls made by scripting functions **963**.

The isolation functions performed by scripting functions **963** strip irrelevant information from a URL's page. Thus, if a page contains a header, some opening text, and a list of celebrity addresses, and the topic associated with the page is celebrity addresses, the scripting functions can isolate the celebrity addresses from the header and the opening text. This is accomplished using the HTML object models, which allow the scripting functions to manipulate the URL's page based on HTML tags in the page. For example, the scripting functions can retrieve all of the text lines associated with <anchors> HTML tags by calling an HTML object method that performs that function.

In most embodiments, the scripting functions treat the text string associated with an individual HTML tag as an individual entry. Thus, if a multi-word text string is associated with an HTML tag, the entire text string is considered one phrase. By storing text strings found on network pages as single units, the present invention improves the precision of the topics it returns. Thus, if "John Glen's Address" appears on a celebrity page and "John's Apple Store" appears on a shopping page, a search query for "John Glen's Address" will only return a hit for the celebrity page and not for the shopping page, even though both pages contain the word "John". If the terms on the pages were stored individually, both pages would produce a hit resulting in an erroneous identification of the topic of the search.

For each entry isolated by scripting functions **963**, WEB-PARSE **962** places the entry and its associated topics in index database **967** and/or HTML files **975**. The entries that

are placed in index database **967** are placed there by making database calls to database server **972** and can be accessed later by making additional calls to database server **972**. The entries that are placed in HTML files **975** are placed there by Topics Dictionary **239** so that they may be indexed by an Index server **973** to form Index server files **974**. Whether an entry and its associated topics are added to the index database or the Index server files is controlled by a property in URL list **960**. The topics for an entry may be any one of or all of the topics listed for this URL page in source database **961**.

At step **603**, WEB-PARSE **962** passes individual terms found in a term list **968** to Topics Dictionary component **239**. Term list **968** includes a list of terms and phrases organized by topics and allows terms to be associated with certain topics in Index database **967** or Index server files **974** even though the terms do not appear in a URL's page. For each term or phrase in term list **968**, database server **972** creates an additional entry in Index database **967** and/or Topics Dictionary **239** creates an additional entry in HTML text file **975** to be indexed by Index server **973**.

In one embodiment of the invention, all forms of the individual terms (e.g. plural, singular, past tense etc.) either from a URL page or the term list are derived before being stored. This derivation creates all of the linguistic forms of each individual term. For example, if the term "televisions" is found in the URL page, it is stemmed to provide both "television" and "televisions".

Once the isolated entries for each of the URL pages listed in source database **961** have been entered in Index database **967** and/or Index server files **974**, the process of FIG. **14A** pauses at step **604** to wait for a search query. When a search query **970** is received, the process continues at step **605** of FIG. **14B** where Topics Dictionary **239** divides the query into sub-queries. Specifically, each quoted phrase in the user's query and each multi-word entry designated by NLP component **227** are set as separate sub-queries. Once the phrase and multi-word entries have been grouped into sub-queries, the remaining terms in the user's query are grouped as a single sub-query.

Each sub-query found above is sequentially processed through the remaining steps of FIG. **14B**. In step **606**, one of the sub-queries is selected and the number of terms in the sub-query is used to initialize a search length variable "N". At step **607**, N is compared to "1" to determine if there is more than one term in the sub-query. If there is more than one term, Topics Dictionary **239** uses database server **972** and/or Index server **973** to search through Index Database **967** and/or Index server files **974** for the entire sub-query at step **609**. If one or more matches are found for the entire sub-query at step **610**, the associated topics **971** are returned to IEXPLORE **224** at step **612**. In one embodiment, the topics are returned in an HTML page. However, those skilled in the art will recognize that the topic may be returned in any suitable format. If a match cannot be found for the entire sub-query at step **610**, the length variable N is reduced by one at step **611** and control returns to step **607**. If N is again greater than one at step **607** the length variable N is reduced by one at step **611** and control returns to step **607**.

If N is again greater than one at step **607**, Topic Dictionary **239** searches for all phrases of length N found in the sub-query. Thus, if the initial query was (A B C D), where A, B, C, and D are each terms, the first reduction in size produces two search units (A B C) and (B C D) that are searched for individually in Index database **967**. If either of these smaller units is found Index database **967** and/or Index server files **974** at step **610**, the associated topics **971** are returned at step **612** and the terms corresponding to the matching phrases are removed from the sub-query. If both of these smaller units are found in Index database **967** and/or Index server files **974** at step **610**, the associated topics **971** for both units are returned at step **612** and all of the terms of the sub-query are removed.

If neither of these smaller units is found in Index database **967** at step **610**, the length variable N is again reduced by one at step **611**. If N is still greater than one at step **607**, Topics Dictionary **609** searches for all phrases of length N found in the search query. Using the example above, this produces three units (A B) (B C) and (C D), which are each searched for in Index database **967**.

Steps **607**, **609**, **610** and **611** are repeated until a query unit is found in Index database **967** or Index server files **974**, or until N equals one.

When N equals one at step **607**, or after topics have been returned at step **612**, the process continues at step **608** where N is reset to equal the number of terms remaining in the sub-query. This number will be less than the number of terms originally in the sub-query if terms were removed in step **612** after a match was found. At step **608**, only those terms that did not have a match are left in the sub-query.

At step **613**, N is compared to one and if it is greater than one, Topics Dictionary **239** places a Boolean "AND" between each term of the sub-query at step **614**. Continuing the example above, this produces a logical search string (A AND B AND C AND D). Topics Dictionary **239** then searches for strings that have each of the terms of the logical search string. Any string that has all of the search string's terms, regardless of the order of the terms in the string, will be considered a match at step **615**. If there are no strings that match the logical search string at step **615**, N is reduced by one at step **616** before being compared to "1" at step **613**.

If N is greater than one at step **613**, step **614** is repeated using only N terms in each combination. Using the example above with N reduced to "3", Topic Dictionary **239** searches based on four logical search strings (A AND B AND C), (A AND B AND D), (A AND C AND D), and (B AND C AND D). If none of these logical search strings result in a match at step **615**, then steps **616**, **613**, **614**, and **615** are repeated until there is a match or until N equals one. If there is a match at step **615**, Topics Dictionary **239** returns the matches to IEXPLORE **224** in a topic list embedded in an HTML page at step **617**. Topics Dictionary **239** also removes the terms of the matching logical search string from the sub-query.

If N is equal to one at step **613**, or after topics are returned at step **617**, the process continues at step **618**, where Topics Dictionary **239** searches for each remaining term of the sub-query on an individual basis. If there is at least one match at step **619**, Topics Dictionary **239** determines if there are fewer matches than a maximum number at step **620**. In the embodiment of FIG. **14B**, the maximum number is twenty but other numbers are possible. If there are fewer than the maximum number of matches, Topics Dictionary **239** returns the corresponding topics at step **621**. If more than one term of the sub-query matches, the corresponding topics are returned for each term.

In one embodiment, the topic lists returned at steps **612**, **617** and **621** include lists of the matching topics, the number of matches for each topic, and weighting bits that indicate if the keywords match an entire string stored in Index database **967** or Index Server Files **974**. Thus, if the user's query is

"Tom Hanks in Saving Private Ryan", and "Saving Private Ryan" is stored under the MOVIE topic, a weighting bit would be returned with the topic MOVIE in the topic list.

After the topic list is returned at step **622** or if there are no matches at step **619** or if there are more than the maximum number of matches at step **620**, Topics Dictionary **239** checks to see if there are more sub-queries to process at step **622**. If there are more sub-queries, the process returns to step **606**. If there are no more sub-queries, the process returns to step **600** of FIG. **14**A to await a new user query.

The progressive reduction in size of the search units described above improves the precision of the topics returned by the present invention. Specifically, since a match on a multi-word part of the query is likely to be more relevant to the query than a match on a single word, the invention is more likely to return relevant topics than if searching was simply performed on the individual terms of the query.

To facilitate a dynamic database that includes the latest news events and latest additions to the network, the present invention periodically returns to step **602** from step **604** to retrieve updated versions of the pages on the network. The refreshed pages are then stripped to build an updated index as discussed above.

Through the processes described above in connection with Topics Dictionary **239**, the present invention is able to automatically generate a list of indexed terms organized under topics. This list includes terms that have recently entered public discourse such as the names of people who were otherwise unknown to the general public one month earlier. This aspect of the present invention allows possible search goals to be identified based on search terms that are new to the public vocabulary. It also allows possible search goals to be identified based on new events. If there's a hurricane named Mitch in Florida and someone types in "hurricane Mitch", the present invention can determine that the searcher might be interested in the recent news on the Florida hurricane.

Returning to FIG. **10**, after the topics have been returned by database server **239** at step **488**, the remaining steps of FIG. **10** are executed thereby finishing step **554** of FIG. **13**. The process of FIG. **13** then continues at step **556** where the topic list returned by database server **239** is stored for later use.

At step **558** of FIG. **13**, QUERYENG **208** generates a topic list based upon the NLP bits produced during the natural language parse of the search query. As noted above, many terms returned in the NLP block are associated with a set of NLP semantic bits or flags. The topics are selected based upon associations between the topics and the NLP semantic bits. Table 1 provides examples of such associations found in embodiments of the present invention.

TABLE 1

| NLP BIT | TOPIC |
|---|---|
| +Nme | First name |
| +Compny | Business |
| +Cty | City |
| +Stte | State |
| +Cntry | Country |
| +Contnt | Continent |
| +Url | URL |
| +Email | E-mail |
| +Wthr | Weather |
| +Food | Food |

Once topics have been identified for the NLP semantic bits returned by the NLP parse, the process of FIG. **13** continues at step **560** where the topics based on Topics Dictionary **239** and the NLP semantic bits are combined to identify the most likely topic for the search. In most embodiments, the topic lists produced by Topics Dictionary **239**, and the NLP semantic bit process include the number of terms from the search query that have been found under each topic. To combine these types of topic lists, the number of hits under similar topics in different topic lists are added together in some embodiments. Thus, if the Topics Dictionary list included three hits under the topic Celebrity, and the NLP semantic bit list included two hits under the topic Celebrity, the combined topic list would include five hits under the topic Celebrity. In other embodiments, the topic lists remain separate and are used as independent clues to determine the user's goal.

When the topics are combined, the combined topic list is sorted at step **562** of FIG. **13**. The sorting places the topic with the most hits at the top of the combined topic list. The combined topic list is then returned at step **564** to complete the process represented by box **328** of FIG. **8**. If the topics are not combined, the individual topic lists are returned at step **564**.

At step **330** of FIG. **8**, QUERYENG **208** designates the keywords as the search terms to be used during searching. By reducing the users search query to just its keywords, the present invention improves the efficiency of the search performed by the selected search service. However, the original search query may also be used during the search.

At step **332**, QUERYENG **208** selects a search area or repository where the search is to be performed. When operating on the searcher's initial search query the search area is a generic search engine. In particular, the search area is the best vector-space search engine available. A vector-space search engine is one that ranks documents on the number of times a term in the search query appears in the document, discounting those terms that are common to a large number of documents and giving priority to documents that have the term in the title or the first paragraph of the document. Each vector-space search engine has its own algorithm for weighting these different factors. In most embodiments, the initial search engine selected at step **332** is stored in Registry **222** of FIG. **2**.

After the search area is selected in step **332**, QUERYENG **208** determines if the search should be converted into a logical search query, also known as a logical function query, at step **334**. Logical searches have logical operators between search terms that indicate the relationship between the terms. Examples of logical operators include "AND" to indicate that two terms should appear together in a document, "OR" to indicate that one of the terms or both of the terms should appear in the document, "NOT" to indicate that a term should not appear in the document, and quotes to indicate that the exact phrase found between the quotes should appear in the document.

For the initial query, a logical search is not constructed since the query is being submitted to a vector-space search engine and such search engines work better if they do not have to deal with logical operators. Later in the process, after

possible search goals have been identified, the determination of whether to construct a logical search query is based largely on the user's search goal. For search goals that involve specific items, the present invention first attempts to locate a search area dedicated to the item. For example, if the user is looking for a celebrity's address, the present invention will look for a search area dedicated to celebrity addresses. If such a search area cannot be found, the present invention will convert the search query into a logical search query and will submit the logical search query to a search engine that supports such queries.

If a search area exists that is targeted at the user's goal, and the search area works better when it receives logical queries, the present invention will convert the query into a logical query. If the search area works better when it receives free text search queries, the query is not converted into a logical search query.

To determine if a particular search area is better suited to receiving logical search queries or free text search queries, QUERYENG **208** accesses a table that indicates the preferred search query form for a number of search areas. In most embodiments, this table is stored in default.htm.

Constructing a logical query is shown as step **336** of FIG. **8** and under embodiments of the present invention this step relies on the NLP data returned as a result of the natural language parse performed in step **324**. The specific process for constructing the logical search is described in the flow diagram of FIG. **15**, which begins with a step **700**. In step **700**, logical operators for the selected search area are identified by QUERYENG **208** because different search areas use different logical operators. For example, in some search areas the logical operator "AND" is represented by "+". QUERYENG **208** keeps a list of the characters and/or strings used by various search areas to represent the logical operators. This provides a significant advantage in the present invention because the user does not need to remember the specific operators used by a search area in order to efficiently use the search area. The present invention automatically inserts the proper operators for the search area.

For simplicity in the discussion below, the invention is described using the Boolean tags: "AND", "OR", "NOT", "NEAR", and "( )". However, those skilled in the art will recognize that the present invention actually inserts the search area's character or string in the logical searches constructed below. Thus, if the term "AND" is used in the discussion below and the selected search area represents "AND" using the "+" character, a "+" will actually be inserted in the logical search instead of the "AND".

In step **702** of FIG. **15**, the NLP data is examined to group terms that are in a coordinating relationship into a coordinating (CRD) set. Terms in a coordinating relationship are connected together by the words "or" or "and". For each CRD set at step **702**, a set of operations is performed as shown in box **704** of FIG. **15**.

The first operation in box **704** is actually a set of operations that are repeated for each CRD term in the CRD set. The operations performed for each CRD term are found in box **706** and the fact that these operations are repeated is represented in box **708**.

The first operation in box **706** is step **710** where the current CRD term is inserted into the Boolean query along with any inflected forms of the CRD term returned by NLP. The CRD term and its inflected forms are separated from each other by "OR", and the complete string is encased in parentheses to produce: (CRD-TERM OR INFLECTED-FORM-#1 OR INFLECTED-FORM-#2 . . . OR INFLECTED-FORM-#N), which is referred to as a base string.

At step **712**, each of the terms in the search query that modify the current CRD term are identified by QUERYENG **208** based on the NLP data. For each modifying term, the operations in block **714** are repeated. Specifically, step **716** of block **714** is repeated where the modifying term is inserted in the Boolean query along with any allowed inflected forms of the modifying term. The modifying term and its allowed inflected forms are separated from each other by logical OR's and the entire string is encased in parentheses to produce:(MODIFYING-TERM OR INFLECTED-FORM-#1 OR INFLECTED-FORM-#2 . . . OR INFLECTED-FORM-#N), which is referred to as a modifying string.

Note that the present invention is able to discriminate between terms that should have their inflected forms included in the Boolean search and terms that should not have their inflected forms included in the Boolean search. Specifically, QUERYENG **208** filters inflected forms of modifying terms that are grammatically incorrect. Thus, a modifying term such as black, as in the phrase "black bird", would not have its inflected form "blacks" included in the Boolean query. In fact, under the present invention, most modifying terms will not have an inflected form included in the Boolean query. In addition, proper nouns, such as Seattle, are not inflected by the present invention. This avoids adding non-existent terms, such as Seattles, to the Boolean query.

In step **718**, either a "NEAR" or an "AND" is placed between each of the modifying strings as well as between the base string and its neighboring modifying strings. The decision between using "NEAR" and "AND" is made on the basis of the phrase strength currently employed by the web companion and on whether the chosen search service supports NEAR. The phrase strength is initially set at a strong default setting that would cause "NEAR" to be inserted between the modifying strings. However, the user may change this phrase strength during the searching process so that "AND" is used instead of "NEAR" by accepting a suggestion from the Web Companion to make this change.

In step **720**, the modifying strings, the base string and their connecting logical operators are encased in parentheses to produce in the default case: (Base-String AND MODIFYING-STRING-#1 . . . AND MODIFYING-STRING-#N), which is identified as a CRD term string.

In step **722**, Boolean operators are placed between each of the CRD term strings. The Boolean operators are selected on the basis of the coordinating relationship of the current CRD set. If the coordinating relationship is based on "or", an "OR" Boolean operator separates each CRD term string. If the coordinating relationship is based on "and", an "AND" Boolean operator separates each CRD term string. Note that after the initial query has been searched, the query may be refined by the present invention by asking the user to clarify the meaning of "and" in the search query. This process is discussed below in connection with FIGS. **35** and **36**. Based on the clarification made by the user, an "OR" Boolean operator may be used between CRD strings that are based on "and".

In step **724**, the CRD term strings and their corresponding Boolean operators are encased in parentheses to produce: (CRD-term-string-#1 AND/OR CRD-term-string-#2 . . . AND/OR CRD-term-string-#N), which is referred to as a CRD set string.

In step **726**, each CRD set string is separated from all other CRD set strings in the Boolean query by inserting the Boolean operator "AND" between each CRD set string.

At step **728**, the remaining stand-alone terms in the search query are added to the Boolean query. This is accomplished by repeating the steps found in box **730** for each stand-alone term. The first step in box **730** is step **732**, which inserts the stand-alone term after the last CRD set string. In addition, step **732** inserts any inflected forms of the stand-alone term. A Boolean "OR" is inserted between the stand-alone term and each inflected form. In step **734**, the stand alone term, its inflected forms, and the Boolean "OR" operators are encased in parentheses to produce: (STAND-ALONE-TERM OR Inflected-form-#1 . . . OR Inflected-form-#N).

At step **736**, the current stand-alone term's respective NLP bits are examined to determine if the term is associated with a +NEG bit. This bit indicates that in the original search query the current term was modified by another term in such a way that it appears the user wants to exclude documents that include the current term. If the current term is not associated with a +NEG bit, a Boolean "AND" is inserted before the open parentheses of the stand-alone string at step **740**. If the current term is associated with a +NEG bit, a Boolean "NOT" is inserted before the open parentheses of the stand-alone string at step **738**. As with CRD relationships, the query may be refined by asking the user to clarify what they intended a negative modifier to signify. This process is discussed further below in connection with FIGS. **33** and **34**. Based on the user's clarification, an "AND" may be used instead of "NOT" even though there is a negative modifier.

Once all of the stand-alone terms have been added, the process of constructing the Boolean search query is complete.

Note that in the discussion above, QUERYENG **208** treats multi-word entries returned by the natural language parse as a single term that is placed in quotes or is otherwise grouped as a Boolean phrase. Thus, the multi-word entry "Microsoft Corporation" would be considered a single term in the discussion above and would appear within phrase markers in the constructed Boolean query.

After the Boolean search has been constructed at step **336** of FIG. **8** or if a Boolean is not to be constructed at step **334**, the process continues at step **338** where QUERYENG **208** determines if the search query is to be modified. Typically, such modifications result from user selections made in response to web companion displays discussed further below. If the query is to be modified at step **338**, the process continues at step **340** where the query is appropriately modified.

After the query is modified in step **340**, or if the query is not to be modified in step **338**, the search query, either logical or free text, is submitted to the selected search area at step **342**. The process of submitting the search to a search area located on the Internet is shown in detail in the flow diagram of FIG. **16**.

In step **760** of FIG. **16**, QUERYENG **208** asks WEB COMPANION **200** to locate or establish a top instance of IEXPLORE, shown as IEXPLORE **218** in FIG. **2**. QUERYENG **208** then generates a URL based on the selected search area and the search query, and passes the URL to WEB COMPANION **200** at step **762**. At step **764**, WEB COMPANION **200** passes the URL to IEXPLORE **218**, which uses the URL to locate the server containing the desired search area and to pass the search query to the search area at step **766**. When the search area completes its search, its respective server returns the search results to IEXPLORE **218** at step **768**.

When the search results are returned to IEXPLORE **218**, they are displayed by IEXPLORE **218** in an independent browser window. This step is represented in FIG. **8** as step **350**.

After the search has been submitted at step **342**, but before the results have been returned, QUERYENG **208** continues to operate at step **344**, where it uses the possible topics determined in step **328**, along with user profile clues, search scope clues and past web companion interactions to determine suggestions to be displayed in the next web companion screen produced by WEB COMPANION **200**, QUERYENG **208**, and DEFAULT.HTM **204**. Thus, each of the items used to identify possible topics as well as user profile clues, search scope clues and past web companion interactions are all features of a clue stream for identifying search suggestions.

The user profile clues include such things as the user's age, their search history, their gender, things they have deemed as favorites, things in their browsing cache and their expertise level. The user profile may be constructed by asking the user for information or by tracking the user's interests based on the searches the user enters or the types of pages the user views.

The search scope clues provide an indication of what level of information the user is seeking. In other words, is the user looking for specific information or just a general overview? In one embodiment, the number of words in the user's initial search text provides a scope clue. Specifically, if the search text includes two or fewer words, the search is considered to have a broad scope. Other criteria for determining scope include broadly worded questions having phrases such as "tell me all about . . . " or "give me information about . . . " that indicate a broad scope. Or narrowly worded questions such as "who was . . . " or "when did . . . ", which indicate a narrow scope. In addition, the user's past searches may be used to determine scope since a searcher is more likely to want detailed information about a subject if they have searched it before.

The past web companion interactions are used to avoid presenting the user with the same suggestions more than once and to provide a logical progression in the types of suggestions provided to the user.

The user profile clues, search scope clues and past web companion interactions each provide interaction characteristics that indicate how the user wants to interact with the web companion. For example, based on the user profile, the web companion can determine whether the user is a child and thus whether the user expects the web companion to interact on a child's level. From the scope clues, the web companion can determine if the user is using the web companion for browsing or to find specific information. From the past web companion interactions, the web companion can identify the types of suggestions that the user is most likely to be interested in seeing next.

In most embodiments, the suggestions that are likely to be most helpful to the user are provided first, with less helpful suggestions provided in later screens. In accordance with this philosophy, some embodiments of the present invention try to provide suggestions based on a users possible search goals first.

To identify possible search goals, the present invention uses a support vector machine (SVM) **209** of FIG. **2** that treats each of the clue stream features as a component of a feature vector also known as a goal vector. The support vector machine compares the query's goal vector to a number of goal surfaces in an n-dimensional goal space defined by n features. Each n-dimensional goal surface separates the goal space into two sections. If a query's goal vector is in one section, the user has the particular search goal associated with the goal surface. If the query's goal

vector is in the other section, the user does not have the particular search goal. For example, a "celebrity" goal surface may divide the goal space into a first section that indicates that the user's goal concerns a celebrity and a second section that indicates that the user's goal does not concern a celebrity. In addition, based on the distances between the query's goal vector and a goal surface, the SVM is able to return the probability that the user has a particular search goal.

The training and use of SVM **209** is shown in the flow diagram of FIG. **17**. The steps required to train SVM **209** are shown in training box **988**. Through these steps, SVM **290** defines the goal vector space and populates it with goal surfaces. The training begins at step **989** where a person manually analyzes a corpus of queries to assign each query to between one and four potential goals. To do this, the person looks at each query and attempts to determine the user's search goal from the query.

The corpus of queries is then submitted to QUERYENG **208** at step **990**. QUERYENG **208** generates a list of features for each query including NLP semantic bits, a list of topics, etc. Advanced embodiments include user profile features associated with the user who generated the training query. For each query, this list of features and the associated potential tasks for that query are then submitted to SVM **209** at step **991**.

SVM **209** generates the goal vector space in step **992** by converting each set of features into a vector in the goal vector space. The resulting goal vector space is then divided by a set of goal surfaces based on the goals identified for each training vector. Techniques for generating these goal surfaces are discussed in greater detail in a pending patent application entitled METHODS AND APPARATUS FOR BUILDING A SUPPORT VECTOR MACHINE CLASSIFIER, filed on Apr. 6, 1998, and having Ser. No. 09/055,477, which is hereby incorporated by reference. In most embodiments, the surfaces are represented by equations that define hyper-planes, which extend through the goal space.

After SVM **209** has been trained, it is ready to be used to identify possible goals of a new search query. The steps involved in using SVM **209** are shown within box **993** of FIG. **17**.

In step **994**, the new search query is submitted to QUERYENG **208**, which identifies a set of features using the techniques described above. The features are submitted to SVM **209** at step **995** and SVM **209** converts the features into the query's goal vector.

At step **996**, SVM **209** determines where the query's goal vector resides in the goal space relative to the goal surfaces. In particular, for each goal surface, SVM **209** determines if the query's goal vector is on the surface's "positive" side indicating that the user's actual search goal is the search goal associated with the surface, or the surface's "negative" side indicating that the user's search goal is not the search goal associated with the surface.

In addition, SVM **209** determines the distance between the query's goal vector and each of the goal surfaces in the goal space. The distance measurement can weight all features equally or can give additional weight to certain features, such as topics

Based on the relative distances between the query's goal vector and each of the goal surfaces, SVM **209** assigns probabilities to each goal. Thus, if the query's goal vector is located next to a number of goal surfaces, there is a low probability that any one goal is the user's actual goal. If the query's goal vector is far from a particular goal surface and is on the positive side of the goal surface, there is a high probability that the associated goal is the user's actual goal.

In step **997**, SVM **209** returns each of the calculated probabilities to QUERYENG **208** for further processing as described below.

In some embodiments, SVM **209** can be trained on a continuing basis using queries entered by the user. This training requires that the user select a goal that is presented to them as shown in step **998**. Based on this selected goal, and the features associated with the user's query, SVM **209** adds an additional corpus goal vector to the vector space at step **999**. Alternatively, SVM **209** can modify an existing corpus goal vector so that it moves closer to the query's goal vector.

The search goal probabilities returned by SVM **209** may also be generated using a simple rules-based engine comprised of a series of complex case statements that test combinations of search clues. The search goal probabilities may also be determined using a Bayes Net.

QUERYENG **208** uses the returned search goal probabilities to select a set of search suggestions. Thus, if there is a high probability that the user is looking for used car prices, QUERYENG **208** will suggest searching a site listing used car prices. If the probability of a search goal is too low, QUERYENG **208** does not make a suggestion based on that goal.

In addition to or instead of providing suggestions based on the possible search goals, embodiments of the invention can also provide scope-based suggestions, which are based almost entirely on scope clues. Examples of scope-based suggestions are shown in FIGS. **28** and **29** discussed further below. Since scope-based suggestions tend to be less helpful than goal-based suggestions, many embodiments will show goal-based suggestions instead of scope-based suggestions if possible. QUERYENG **208** can also provide a suggestion to fine-tune the search query or to select a different search engine. However, since these suggestions are not as helpful, they are usually presented only if other suggestions cannot be made.

Once QUERYENG **208** has determined the suggestions it will display, the process continues at step **346** where the character's behavior changes or is modified based on the suggestions being displayed. For example, if suggestions relating to travel are displayed, the character can be modified so that it appears in travel clothes. Examples of such modifications to the character are described below in connection with the examples of screens displayed by the present invention.

At step **348**, the next web companion screen is displayed, which contains text and control buttons that appear within a balloon produced by WEB COMPANION **200**. The text and control buttons are produced by QUERYENG **208** and IE4 control **202**, respectively. The screen also includes an animated character produced by Search Agent **206** of FIG. **2**, which in one embodiment is implemented through Microsoft AgentX technology. Note that the web companion screen appears at around the same time that the search results from the last search are displayed in a browser window by IEXPLORE **218**. Examples of the web companion screens are shown in FIGS. **18**, **19**, **20**, **21**, **22**, **23**, and **24**, which are each discussed below.

In FIG. **18**, the present invention provides a web companion screen based on a search query that included the terms East Africa and Kenya. Based on these terms, QUERYENG **208** has identified possible topics of country and continent leading to possible goals **802**, **804**, **806**, and **808** of planning a trip, booking a trip using Expedia, finding cheap flight information, and gathering general information about the continent of Africa, respectively. Note that in many embodiments these goals are shaped in part by the user's profile. If the user is only ten years old, the search goals

would be limited to obtaining information about Africa since it is unlikely that a ten year old will be booking a trip to Africa.

In FIG. **18** the animated character **800** has been modified in step **346** of FIG. **8** in light of the displayed suggestions. In particular, character **800** is wearing a hat and carrying a camera to reflect the travel related suggestions in balloon **810**.

Balloon **810** also includes a suggestion **812** that allows the user to indicate that they have found what they were looking for. If the user indicates that their search was successful by selecting suggestion **812**, QUERYENG **208** makes a record of the user's search query and the final URL that produced the results the searcher wanted. In one embodiment, this record is kept in registry **222**, but in other embodiments may be kept in any suitable memory location. QUERYENG **208** accesses this record each time a new search query is entered by the user so that it can determine if the user has made this search before. If it finds a record of a successful result for this search, QUERYENG **208** will suggest to the user that they use this past result. In other embodiments, this is implemented in SVM **209** by adding the URL as a corpus goal with the features associated with the search query forming the corpus goal vector. In further embodiments of the invention, QUERYENG **208** keeps track of the number of times the user selects this past result. If the number of times is greater than some chosen threshold, QUERYENG **208** automatically displays the result without making the suggestion to the user.

In FIG. **19**, the user's search includes terms related to food and based on probabilities from SVM **209**, QUERYENG **208** has identified possible search goals **814** and **816** that relate to recipes. QUERYENG **208** has also caused SEARCH-AGENT **206** to modify animated character **818** so that it is wearing a chef's hat.

FIG. **20** shows a display based on a user query that included a person's name. Although the name topic has been identified, the name did not trigger the celebrity topic. As such, the SVM has determined that the displayed suggestions should be focused on possible search goals a searcher may have relative to a non-famous person. These possible goals include wanting the person's e-mail address (suggestion **822**), the person's mail address (suggestion **824**), and the person's home page (suggestion **826**).

In FIG. **21**, the user's search text also included a person's name. However, the person's name was either indexed by URL index **241** or database server **239** of FIG. **2** as being a celebrity name. Based on the celebrity topic returned by one of these index components, along with other features, QUERYENG **208**, using SVM **209** has provided a different set of suggestions from the suggestions shown in FIG. **20**. Specifically, FIG. **21** includes suggestions **832**, **834**, **836**, **838**, **840**, and **842** that respectively suggest, going to the most popular site concerning the celebrity, searching for photos and images of the celebrity, finding sound files of the celebrity, finding biographical information about the celebrity, finding everything possible about the celebrity, and seeing what CINIMANIA has to say about the celebrity.

In FIG. **22**, the search entered by the user included a business name that produces a hit for the topic BUSINESS. Based on this topic, and other features, QUERYENG **208** determined that the user may be interested in the business's homepage (suggestion **848**), the business's address or phone number (suggestion **850**), or public info about the business, such as stock quotes (suggestion **852**).

In FIG. **23**, the user's search appears to have included a URL. In response, QUERYENG **208** suggests going to the web site represented by the URL (suggestion **856**), and finding web site's that reference the URL (suggestion **858**).

FIG. **24** shows a display of the present invention produced by QUERYENG **208** in response to a query that includes a city name. Since city names trigger an NLP bit to be produced by NLP component **227** of FIG. **2**, QUERYENG **208** is able to identify "city" as a possible topic of the search. As such, QUERYENG **208** produces suggestions that include possible goals related to the topic "city". These suggestions include looking at an entertainment site for the name of the city (suggestion **859**), looking in the cities yellow pages (suggestion **861**), booking a flight to the city using Expedia (suggestion **863**), obtaining cheap flight info (suggestion **865**), and searching for historical information from the Library of Congress (suggestion **867**).

FIG. **25** shows a display triggered by a hit for a movie/restaurant topic. Based on this topic, QUERYENG **208** suggests looking at a web site that is focused on a cities local arts and entertainment, (suggestion **860**) and looking at the yellow pages (suggestion **862**).

FIGS. **26** and **27** provide examples of context-based or media type suggestions found in displays produced when QUERYENG **208** is able to identify possible contexts or media types that the user may be looking for. On the Internet, files come in a wide range of media types including sound, video, picture, and text. In FIG. **26**, based on a topic hit from category index **239** of FIG. **2**, QUERYENG **208** has determined that the user is looking for a sound file. To find this media type, QUERYENG **208** suggests looking in two sites, BillyBoy's sound search (suggestion **868**) and Make-Waves (suggestion **870**). In addition, at suggestion **872**, QUERYENG **208** suggests modifying the search text to include terms like ".wav". In FIG. **27**, the user's search included a term that category index **239** placed under the topic "pictures". Based on this topic, QUERYENG **208** suggests searching for a picture in two different search areas: Binco's picture search (suggestion **874**) and Plish's image surfer (suggestion **876**).

FIG. **28** provides an example of scope based suggestions. In particular, the display of FIG. **28** shows suggestions provided when QUERYENG **208** has identified that the search has a narrow scope. In other words, that the user wants specific, detailed information. Normally, the display of FIG. **28** is only provided if a possible goal could not be identified based on the search or if the user did not select one of the offered goals in an earlier screen. The display includes suggestions for other search areas that provide specific information such as Microsoft's Encarta online encyclopedia (suggestion **878**), online dictionaries (suggestion **880**) and maps (suggestion **881**), and the Internet Public Library (suggestion **882**).

FIG. **29** also provides scope based suggestions, except that the suggestions found in FIG. **29** are for a search having a broad scope, where the user wants general information. The suggestions include going to a site that has common question and answers, known as a Frequently Asked Question (FAQ) site, going to the most popular site for the search term in the query, going to a newsgroup about the search term, and going to a site that has been rated by editors as the best site for the search term (suggestion **883**). In addition, suggestion **884** of FIG. **29** suggests obtaining an overview of the search terms.

FIG. **30** is a display that provides suggestions such as fine-tuning the search (suggestion **888**) and trying a new search service (suggestion **889**). In some embodiments, the display of FIG. **30** is only shown if QUERYENG **208** could not identify possible goals or scope based suggestions or if the user did not select any of the presented goals or scope based suggestions found in previous screens presented to the user.

Depending on what the user selects from the displayed screen, the process of FIG. **8** continues along different paths.

For example, if the user selects option **851** of FIG. **22** or option **889** of FIG. **30**, thereby indicating that they want to perform a new search, the process continues at step **320** of FIG. **8** where QUERYENG **208** solicits the user for the new search text. The selection of the new search option is shown in FIG. **8** by NEW SEARCH box **352**, which provides a path to step **320**.

If the user selects the exit option in any of the screens, WEB COMPANION **200** closes all of the modules that it has invoked except IEXPLORE and closes itself. This option is indicated in FIG. **8** by EXIT box **354**, which provides a path to end state **356**.

If the user selects any other suggestion such as a search goal, a context based suggestion, a scope based suggestion, a suggestion to use a different search service, or a suggestion to fine-tune the search query, the process continues at step **358** where the user's selection is recorded for later use in step **344** to determine future screens to be displayed. The selections that lead to step **358** are shown representatively as GOALS box **360**, CONTEXT BASED box **362**, SCOPE BASED box **364**, DIFFERENT SEARCH SERVICE box **366**, and FINE-TUNE box **368**. These boxes are shown only to illustrate some of the possible suggestions that may be provided to the user. Other suggestions are possible within the scope of the invention.

After the user's selection has been recorded in step **358**, QUERYENG **208** determines at step **370** if the user's selection requires additional screens to be displayed before a new search can be submitted or before the present search can be submitted to a new search area. If additional screens need to be displayed, QUERYENG **208** displays those screens at step **372**.

One suggestion that can lead to additional screens is suggestion **888** of FIG. **30**, which suggests fine tuning the search query. If suggestion **888** is selected, QUERYENG **208** determines if the existing query includes any ambiguities. For each ambiguity it detects in the search query, QUERYENG **208** provides a disambiguation screen that requests additional information to disambiguate the query. Examples of such disambiguation screens are shown in FIGS. **32**, **34**, and **36**.

FIG. **32** shows a disambiguation screen used to remove an ambiguity as to time. An example of a query that includes an ambiguity as to time is shown in FIG. **31**. That query states "I want recent articles on Microsoft word." This query is ambiguous as to time because it is not clear what the user means by "recent". QUERYENG **208** detects this ambiguity because the term "recent" receives an NLP semantic bit of "+tme" that indicates that the term relates to time. Based on this NLP bit and the user's desire to fine tune their query, QUERYENG **208** produces display **900** of FIG. **32**, which provides a selectable list of options designed to clarify what the user means by the word "recent". For example, entry **902** in display **900** would restrict the search to pages that are less than thirty days old. Entry **904** would restrict the search to pages that are six months to one year old.

FIG. **33** provides a second example of an ambiguity in a search query. The search query in FIG. **33** is "Why do men lose their hair and not women?" This query is ambiguous in a Boolean sense because it includes the word "not". In Boolean queries, "not" causes many search engines to exclude pages that contain the word following the "not". In the context of the query of FIG. **33**, a Boolean based search engine would exclude pages that have the word "women". QUERYENG **208** identifies this ambiguity on the basis of an NLP bit, known as the "+neg" bit, that is returned by NLP component **227** in response to the presence of "not" in the search query. To clarify whether the user meant to exclude pages that have the word "women", QUERYENG **208** generates display **910** of FIG. **34**. Display **910** provides the user with a choice of excluding pages that have the word "women" or not excluding pages that have the word "women".

FIG. **35** shows a third example of a search query with an ambiguity in it. Search query **916** in FIG. **35** is "I want information on skiing and snow-mobiling in Wyoming." This search is ambiguous because of the word "and" in the query. Most Boolean based search engines would interpret this query as requiring that each returned page include both the term "skiing" and the term "snow-mobiling". However, a user that inputs such a search query typically wants information on "skiing" OR "snow-mobiling". This type of ambiguity is flagged by NLP component **227** in the NLP data returned for the terms "skiing" and "snow-mobiling". Specifically, NLP component **227** places these terms in the same coordinating (CRD) set and indicates that they are joined by the term "and". When determining if the search includes ambiguities, QUERYENG **208** looks for such coordinating sets and provides a disambiguation display, such as display **920** of FIG. **36**, for each such coordinating set. In display **920**, the user is asked whether they intended to find sites on either skiing or snow-mobiling, or intended to find pages that had both skiing and snow-mobiling.

If the user chooses to fine tune their search and there are no ambiguities in the search query, QUERYENG **208** generates display **930** shown in FIG. **37**. Display **930** includes suggestions to use a different search engine, see a list of past searches, add, delete or change words, exclude words from the search, restrict the search to a range of dates, directly edit the Boolean query, see synonyms and hypernyms of terms in the search query, and change the phrase strength of the constructed Boolean. As noted above, the phrase strength of the Boolean determines whether modifying terms are connected to the terms they modify by a Boolean "AND" or a Boolean "NEAR". Many of the suggestions shown in display **930** will lead to additional displays to solicit the specific information. For example, if the user wants to exclude a term, an additional display is presented to ask the user what term they want excluded.

After QUERYENG **208** has collected the additional information it needs to construct and submit a new search query, or if QUERYENG **208** did not need additional information, the process returns to step **332** where a search area is selected. The search area selected on return to step **332** is chosen in large part on the basis of the suggestion selected by the user. For example, each search goal suggested to the user is usually associated with a specific search area. Thus, if the user has selected a suggested search goal, QUERYENG **208** is able to directly identify a search area associated with that search goal.

The associated search area is often focused on providing information related to the search goal. For instance, suggestion **822** of FIG. **20** suggests a search goal of finding a person's email address. This search goal is associated with a search service that is dedicated to storing and searching through email addresses. Similarly, suggestion **814** of FIG. **19**, which suggests the search goal of seeing what Chef BillG has for recipes, has Chef BillG's recipe page as its search area.

Search areas are also associated with context-based suggestions (also known as media type suggestions) and scope-based suggestions. Thus, if the user selects context-based suggestion **868** of FIG. **26**, which suggests using BillyBoy's sound search to find sound files, QUERYENG **208** will select BillyBoy's sound search as the search area. Similarly, if the user selects scope-based suggestion **883** of FIG. **29**, which suggest looking at the best sites about a topic, QUERYENG **208** will select a search engine that reviews all of the sites it includes in its database. Additionally, if the user selects a suggestion to look at an overview of a topic,

QUERYENG **208** selects a search area that has excellent topic overview information.

In addition, if the user has adopted a suggestion to change their search service, QUERYENG **208** will select the search area based on the new search service chosen by the user.

Under the present invention, the user does not have to be familiar with the search area or the information it provides in order to utilize it. The user only needs to select a suggestion that they believe will advance their search. For example, if a user selects a suggestion to find a user's email address, they do not need to know about the email search area the present invention will search. In addition, since many of the suggestions are associated with relatively obscure search areas, the present invention allows users to utilize a broader range of search areas than they would otherwise use.

After the search area has been selected, QUERYENG **208** determines if a logical query should be constructed based on the selected search area and the present form of the search query. If the search query is already in logical form or if the search area works better with free text searches, a logical query would not be constructed.

After the logical search is constructed at step **336** or if at step **334** it is determined that a logical search query will not be constructed, QUERYENG **208** determines if the query should be modified. The modification of the query is based largely on the suggestion selected by the user. For example if the user has selected scope-based suggestion **884** of FIG. **29**, which suggests looking at an overview of a topic, the search query is modified to include terms such as "overview" and "official site". By adding these terms, QUERYENG **208** improves the likelihood that the search query will return pages of a general scope.

The search query is also modified if the user selects certain context-based suggestions, such as suggestion **872** of FIG. **26**. In suggestion **872**, the user is asked if they are interested in adding words to the query that relate to sound files. The selection of this suggestion causes QUERYENG **208** to modify the query to include these sound file terms.

Certain search goal suggestions also lead to modification of the search query. For example, if the search goal is to find information on antique cars, QUERYENG **208** diplays a suggestion to add words like automobile, auto, and classic to the query.

The modification of the query can be more sophisticated than just adding terms. For example, if the user has fine tuned a search query that included an ambiguity, QUERYENG **208** can modify the search query to remove the ambiguity. Thus, QUERYENG **208** can change the coordinating relationship between two words from "AND" to "OR" and can change a connecting term such as "NOT" to "AND" to reflect the user's true searching intentions. In addition, if the search query included an ambiguity as to time, such as including the term "recent", QUERYENG **208** can replace "recent" with a specific set of dates. For certain search areas, ones that include date range fields, QUERYENG **208** removes the term "recent" from the search query and adds instructions in the query to fill out the date range field of the search area with the date range selected by the user. By removing the term "recent" from the search query, the present invention keeps the search area from looking for pages that include the term "recent" and instead focuses the search area on looking for pages that were produced on certain dates. This better reflects the user's searching intention.

The process shown in the flow diagram of FIG. **8** continues to repeat the cycle of presenting search suggestions in parallel with search results, recording the user's suggestion, selecting a search area, constructing and/or modifying a search and submitting the search to the search area until the user elects to exit the web companion program or start a new search. By keeping track of past web companion interactions, the present invention is able to present new suggestions to the user at each cycle, in a manner similar to the way a friend or librarian would suggest alternative searching techniques.

In FIG. **2**, WEB COMPANION **200**, IE4 control **202**, SEARCH-AGENT **206**, QUERYENG **208**, and SPELLCHECK **221** are shown on a client **199**, and NLP component **227**, Topics Dictionary **239** are shown on a server **233**. However, those skilled in the art will recognize that all of the components could appear on client **199** together. Furthermore, those skilled in the art will recognize that QUERYENG **208** could appear on server **233** along with NLP component **227**, and Topics Dictionary **239**. The particular configuration chosen, while affecting performance, is not critical to the basic operation of the invention.

Although the present invention has been described with reference to specific embodiments, workers skilled in the art will recognize that changes may be made in form and detail without departing from the spirit and scope of the invention.

What is claimed is:

**1**. A method of aiding a user in searching a computer environment comprising:

(a) receiving a search query from a user;

(b) generating a clue stream comprised of features based in part on the search query;

(c) providing the clue stream to a support vector machine trained to identify a search goal from a clue stream; and

(d) identifying a search area within the computer environment based on a search goal produced by the support vector machine in response to the clue stream.

**2**. The method of claim **1** wherein generating a clue stream comprises generating at least one topic from a natural language parse of the search query.

**3**. The method of claim **2** wherein generating a clue stream further comprises generating at least one topic by comparing the search query to text found in pages on a network.

**4**. The method of claim **3** wherein generating a clue stream further comprises generating at least one user profile feature.

**5**. The method of claim **4** wherein generating a clue stream further comprises generating a scope feature based in part on the search query.

* * * * *